Calhoun: The NPS Institutional Archive

DSpace Repository

Theses and Dissertations | 1. Thesis and Dissertation Collection, all items

1987-12

SCMOS silicon compiler organelle design and insertion

Baumstarck, Joan Eichten; Kirk, Donald E.

http://hdl.handle.net/10945/22466

Downloaded from NPS Archive: Calhoun

http://www.nps.edu/library

Calhoun is the Naval Postgraduate School's public access digital repository for research materials and institutional publications created by the NPS community. Calhoun is named for Professor of Mathematics Guy K. Calhoun, NPS's first appointed -- and published -- scholarly author.

Dudley Knox Library / Naval Postgraduate School
411 Dyer Road / 1 University Circle
Monterey, California USA 93943

NAVAL POSTGRADUATE SCHOOL
Monterey, California

THESIS

324045

SCMOS SILICON COMPILER
ORGANELLE DESIGN AND INSERTION

by

Joan Eichten Baumstarck

December 1987

Co-Advisor: D. E. Kirk
Co-Advisor: H. H. Loomis, Jr.

Approved for public release; distribution is unlimited

T238689

REPORT DOCUMENTATION PAGE

1a. REPORT SECURITY CLASSIFICATION UNCLASSIFIED	1b. RESTRICTIVE MARKINGS
2a. SECURITY CLASSIFICATION AUTHORITY	3. DISTRIBUTION / AVAILABILITY OF REPORT Approved for public release; distribution is unlimited
2b. DECLASSIFICATION / DOWNGRADING SCHEDULE	
4. PERFORMING ORGANIZATION REPORT NUMBER(S)	5. MONITORING ORGANIZATION REPORT NUMBER(S)
6a. NAME OF PERFORMING ORGANIZATION Naval Postgraduate School 6b. OFFICE SYMBOL (If applicable) 62	7a. NAME OF MONITORING ORGANIZATION Naval Postgraduate School
6c. ADDRESS (City, State, and ZIP Code) Monterey, California 93943-5000	7b. ADDRESS (City, State, and ZIP Code) Monterey, California 93943-5000
8a. NAME OF FUNDING / SPONSORING ORGANIZATION 8b. OFFICE SYMBOL (If applicable)	9. PROCUREMENT INSTRUMENT IDENTIFICATION NUMBER
8c. ADDRESS (City, State, and ZIP Code)	10. SOURCE OF FUNDING NUMBERS PROGRAM ELEMENT NO / PROJECT NO / TASK NO / WORK UNIT ACCESSION NO

11. TITLE (Include Security Classification)

SCMOS SILICON COMPILER ORGANELLE DESIGN AND INSERTION (U)

12. PERSONAL AUTHOR(S)
BAUMSTARCK, Joan Eichten

13a. TYPE OF REPORT	13b. TIME COVERED	14. DATE OF REPORT (Year, Month, Day)	15. PAGE COUNT
Master's Thesis	FROM _____ TO _____	1987 December	86

16. SUPPLEMENTARY NOTATION

17. COSATI CODES			18. SUBJECT TERMS (Continue on reverse if necessary and identify by block number)
FIELD	GROUP	SUB-GROUP	MacPitts; Silicon Compiler; SCMOS; VLSI

19. ABSTRACT (Continue on reverse if necessary and identify by block number)

Conversion of the MacPitts Silicon Compiler from NMOS to Scalable CMOS technology includes the custom design of SCMOS organelles to replace the NMOS organelles in the data-path. Three arithmetic organelles, the incrementor, decrementor and subtractor, are designed using the Magic layout editor. Cell insertions are made to the MacPitts Silicon Compiler installed on ISI workstations.

20. DISTRIBUTION / AVAILABILITY OF ABSTRACT ☒ UNCLASSIFIED/UNLIMITED ☐ SAME AS RPT ☐ DTIC USERS	21. ABSTRACT SECURITY CLASSIFICATION UNCLASSIFIED
22a. NAME OF RESPONSIBLE INDIVIDUAL H. H. LOOMIS, JR.	22b. TELEPHONE (Include Area Code) (408) 646-3214 22c. OFFICE SYMBOL 62Lm

DD FORM 1473, 84 MAR 83 APR edition may be used until exhausted
All other editions are obsolete

GPO Government Printing Office 1986—606-243

Approved for public release; distribution is unlimited.

SCMOS Silicon Compiler Organelle Design and Insertion

by

Joan Eichten Baumstarck
Lieutenant, United States Navy
B.S., University of Florida, 1981

Submitted in partial fulfillment of the
requirements for the degree of

MASTER OF SCIENCE IN ELECTRICAL ENGINEERING

from the

NAVAL POSTGRADUATE SCHOOL
December 1987

ABSTRACT

Conversion of the MacPitts Silicon Compiler from NMOS to Scalable CMOS technology includes the custom design of SCMOS organelles to replace the NMOS organelles in the data-path. Three arithmetic organelles, the incrementor, decrementor and subtractor, are designed using the Magic layout editor. Cell insertions are made to the MacPitts Silicon Compiler installed on ISI workstations.

Thesis
B2243
c.1

TABLE OF CONTENTS

I. INTRODUCTION 9
A. CONVERSION OF MACPITTS SILICON COMPILER 9
1. Previous Thesis Work 9
2. Current Effort 10
B. THESIS CONTENTS 10
1. Chapters 10
2. Appendices 10

II. ORGANELLE DESIGN 11
A. LAYOUT PHILOSOPHY 11
1. Internal Layout 11
2. Bit Slice / Unit Slice Approach 11
B. ORGANELLE DESIGN 12
1. Organelle Summary 12
2. Design of an Organelle 13
3. Logic Simulation 17
C. SPICE SIMULATION 17
D. ORGANELLES AND TEST RESULTS 20

III. ORGANELLE INSERTION INTO MACPITTS 21
A. THE MACPITTS ORGANELLE 21
1. Call and Layout of an Organelle 21
2. Modifications Required to Insert an Organelle 22
B. MODIFYING THE LIBRARY FILE 22
C. MODIFYING THE ORGANELLE FILE 22
D. GENERATING THE MACPITTS LAYOUT 22
1. The Input Program 23
2. Running MacPitts 23
E. COMPARISON OF SCMOS AND NMOS LAYOUTS 24

1. Organelles 24
2. Chip Layouts 25

IV. THE MACPITTS ISI ENVIRONMENT 29
A. STATUS OF THE ENVIRONMENT 29
B. THE CURRENT MACPITTS ENVIRONMENT 30

V. CONCLUSION 33
A. REVIEW OF THESIS WORK 33
1. Scope of Thesis 33
2. Long Learning Curve 33
B. STATUS OF THE MACPITTS SILICON COMPILER 33
1. SCMOS Organelles 33
2. Conversion Efforts 34

APPENDIX A: SPICE SIMULATION 35

APPENDIX B: ORGANELLE DESIGNS 43

APPENDIX C: MACPITTS CHIP LAYOUTS 58

APPENDIX D: SCMOS ORGANELLE INSERTION 70

LIST OF REFERENCES 82

BIBLIOGRAPHY 84

INITIAL DISTRIBUTION LIST 85

LIST OF TABLES

1. MACPITTS ARITHMETIC FUNCTIONS 14
2. TRUTH TABLE 15
3. SPICE GENERATED TIMING DATA 19
4. ORGANELLE COMPARISONS 25
5. ALTERED MACPITTS FILES 30
6. FILE DIRECTORY 31
7. MOSIS TRANSISTOR PARAMETERS 41
8. DECREMENTOR TRUTH TABLE 44
9. DECREMENTOR TIMING DATA 44
10. INCREMENTOR TRUTH TABLE 48
11. INCREMENTOR TIMING DATA 48
12. SUBTRACTOR TRUTH TABLE 52
13. SUBTRACTOR TIMING DATA 53
14. SUBTRACTOR TIMING DATA - VERSION 1 56

LIST OF FIGURES

2.1 MacPitts Internal Layout 12
2.2 Organelle Layout 13
2.3 Logic Expressions 15
2.4 Transistor Schematic (Incrementor) 16
2.5 Data-path Dimensions 18
3.1 Gen-form Format 23
3.2 MacPitts Input Program (incrementor.mac) 24
3.3a SCMOS Organelle Size Measurement 26
3.3b NMOS Organelle Size Measurement 27
3.4 NMOS and SCMOS Taxi Meter Chips 28
A.1 SPICE Simulation Model 39
A.2 Test Organelle Stipple Plot With Node Numbers 40
A.3 Sample SPICE Deck 42
B.1 Decrementor Transistor Schematic 43
B.2 Decrementor Bit 0 45
B.3 Decrementor Bit n 46
B.4 Incrementor Transistor Schematic 47
B.5 Incrementor Bit 0 49
B.6 Incrementor Bit n 50
B.7 Subtractor Transistor Schematic 51
B.8 Subtractor Bit 0 54
B.9 Subtractor Bit n 55
B.10 Subtractor Bit n - Version 1 (not used) 57
C.1 NMOS Decrementor Chip (One Bit Word) 58
C.2 SCMOS Decrementor Chip (One Bit Word) 59
C.3 NMOS Decrementor Chip (Four Bit Word) 60
C.4 SCMOS Decrementor Chip (Four Bit Word) 61
C.5 NMOS Incrementor Chip (One Bit Word) 62

C.6 SCMOS Incrementor Chip (One Bit Word) 63
C.7 NMOS Incrementor Chip (Four Bit Word) 64
C.8 SCMOS Incrementor Chip (Four Bit Word) 65
C.9 NMOS Subtractor Chip (One Bit Word) 66
C.10 SCMOS Subtractor Chip (One Bit Word) 67
C.11 NMOS Subtractor Chip (Four Bit Word) 68
C.12 SCMOS Subtractor Chip (Four Bit Word) 69
D.1 Sample L5 Code for Incrementor Organelle 76
D.2 Format of Library File Entry 77
D.3 Incrementor Library File Entry 78
D.4 Connection Point Measurements 79
D.5 MacPitts Input Program Format 80
D.6 MacPitts Input Programs for the Incrementor 80
D.7 MacPitts Generated Incrementor Chip 81

I. INTRODUCTION

A. CONVERSION OF MACPITTS SILICON COMPILER

The MacPitts Silicon Compiler translates the low level functional description of a circuit into a computer readable mask level description which can then be sent to a vendor for fabrication. It was developed at the Massachusetts Institute of Technology Lincoln Laboratories in the early 1980's. It is a fixed floor plan compiler, with an N-type Metal Oxide Silicon (NMOS) technology dependent design, based on the graphics layout editor Caesar. The effort of the Silicon Compiler Group at the Naval Postgraduate School involves the conversion of the NMOS based MacPitts to Scalable Complementary Metal Oxide Silicon (SCMOS) technology. SCMOS technology has the advantages of smaller feature size and lower power consumption.

1. Previous Thesis Work

The conversion of the MacPitts Silicon Compiler from NMOS technology to SCMOS technology is an ongoing project. The majority of the completed theses laid the groundwork by exposing the inner workings of MacPitts. Prior to their completion, very little documentation existed. Therefore, the completed theses serve as guides for future conversion efforts.

The early work included the installation of the MacPitts Silicon Compiler on the VAX-11/780 by D. Carlson, 1984 [Ref. 1], exposure of the building blocks of MacPitts by A. Froede, 1985 [Ref. 2], and the relationship of the MacPitts input program to the generated chip layout by R. Larrabee, 1985 [Ref. 3]. In 1986, M. A. Malagon-Fajar completed a study of the MacPitts compiler organization and layout language [Ref. 4], and E. Weist developed a flowcharting system and compiler interface for MacPitts [Ref. 5].

With these theses as references, work could begin on changing the actual circuit layouts from NMOS technology to SCMOS technology. The broad area of knowledge required to modify MacPitts code, prompted the initial efforts of SCMOS organelle design and organelle insertion to be completed as separate theses. A. Mullarky designed the first SCMOS organelles in 1987 [Ref. 6], while E. Malagon further exposed the inner workings of MacPitts and successfully inserted SCMOS organelles [Ref. 7]. This work was completed on the Computer Science Department

VAX-UNIX (CS-VAX) computer system. Paralleling these efforts was the work by J. Harmon [Ref. 8] which included the installation of MacPitts on the ISI (Integrated Solutions, Inc.) workstations in the VLSI (Very Large Scale Integration) design lab.

2. Current Effort

The current effort involves continued SCMOS organelle design and insertion, enhancement of the ISI MacPitts environment, SCMOS simulation and routing optimization. The completed theses provided sufficient information for a single thesis student to design and insert SCMOS organelles. This thesis was the first effort of this type. The purpose of this thesis was to design the three remaining arithmetic organelles and insert them into the ISI MacPitts environment. This required inserting the previously designed SCMOS organelles (inserted on the CS-VAX MacPitts environment) into the ISI MacPitts environment, while incorporating the code changes, made by J. Harmon [Ref. 8], which are necessary for MacPitts to run on the ISIs.

B. THESIS CONTENTS

1. Chapters

The main Chapters (II and III) parallel the design and insertion process. Chapter II discusses the SCMOS organelle design phase. A brief description is given of the layout philosophy and the design process is traced for the incrementor organelle. Chapter III covers the insertion of the organelles into the ISI MacPitts environment. The NMOS and SCMOS organelles, and their corresponding chip layouts are compared. Chapter IV gives the status of the MacPitts ISI environment. A list is provided of the files created and modified during the thesis work, along with a brief description of their contents. Chapter V reviews the thesis work and presents the status of the MacPitts Silicon Compiler conversion.

2. Appendices

The Appendices contain the organelle designs, simulation data, chip layouts, and simulation and insertion procedures. Appendix A is a detailed description of the simulation using SPICE, a general purpose circuit simulation program. Appendix B contains the final designs and their simulation data. The chip layouts for the inserted organelles are shown in Appendix C, along with the corresponding NMOS chips. Appendix D contains a step by step guide for organelle insertion. Appendices A and D are provided as an aide for the design and insertion of the remaining SCMOS organelles.

II. ORGANELLE DESIGN

A. LAYOUT PHILOSOPHY

1. Internal Layout

The internal layout of a MacPitts generated chip is composed of a data-path, a flag section and a controller as shown in Figure 2.1. It is within the data-path that the MacPitts default library functions reside. The term organelle is used to represent "the actual layout of a single bit of a function" [Ref. 9: p. 25]. The placement of the organelles within the data-path directs their design, i.e., orientation, dimension, and the placement of ground (GND), power (Vdd), inputs, outputs and carries. A complete description of the MacPitts design organization is presented by E. Malagon [Ref. 7: p. 21]. An overview of the MacPitts organelle layout is given below.

2. Bit Slice / Unit Slice Approach

Each organelle represents one bit of a particular arithmetic function, logical function (boolean and integer), shift function or comparison test. For an n-bit function, n organelles are stacked in a vertical column of the data-path. For example, to increment a four-bit word, four one-bit incrementor organelles are stacked as shown in Figure 2.1.

Each column within the data-path represents one unit, the length of which is determined by the word-length stated in the MacPitts input program (the .mac file, read 'dot' mac). Each horizontal row is one bit slice of the data-path. This arrangement dictates design guidelines for the placement of the inputs, outputs and carries of SCMOS organelles. The carries are referred to as daisies, as the carry outputs are 'daisy' chained down to become the input of the succeeding bit.

Figure 2.2 shows the design layout of the organelles. Bit 0 of the unit is the uppermost organelle (the least significant bit), with bits 1 through n-1 below. The geometric guidelines of the organelles are based on an X Y coordinate system with the origin (0,0 point) at the bottom left corner of the organelle. The guidelines are:

1. Vdd and GND run vertically through the organelle.
2. Clocked circuits have clock lines running parallel to GND and Vdd.
3. Inputs enter on the left.
4. Outputs exit from the top of the organelle, or exit from the right to be extended to the top.

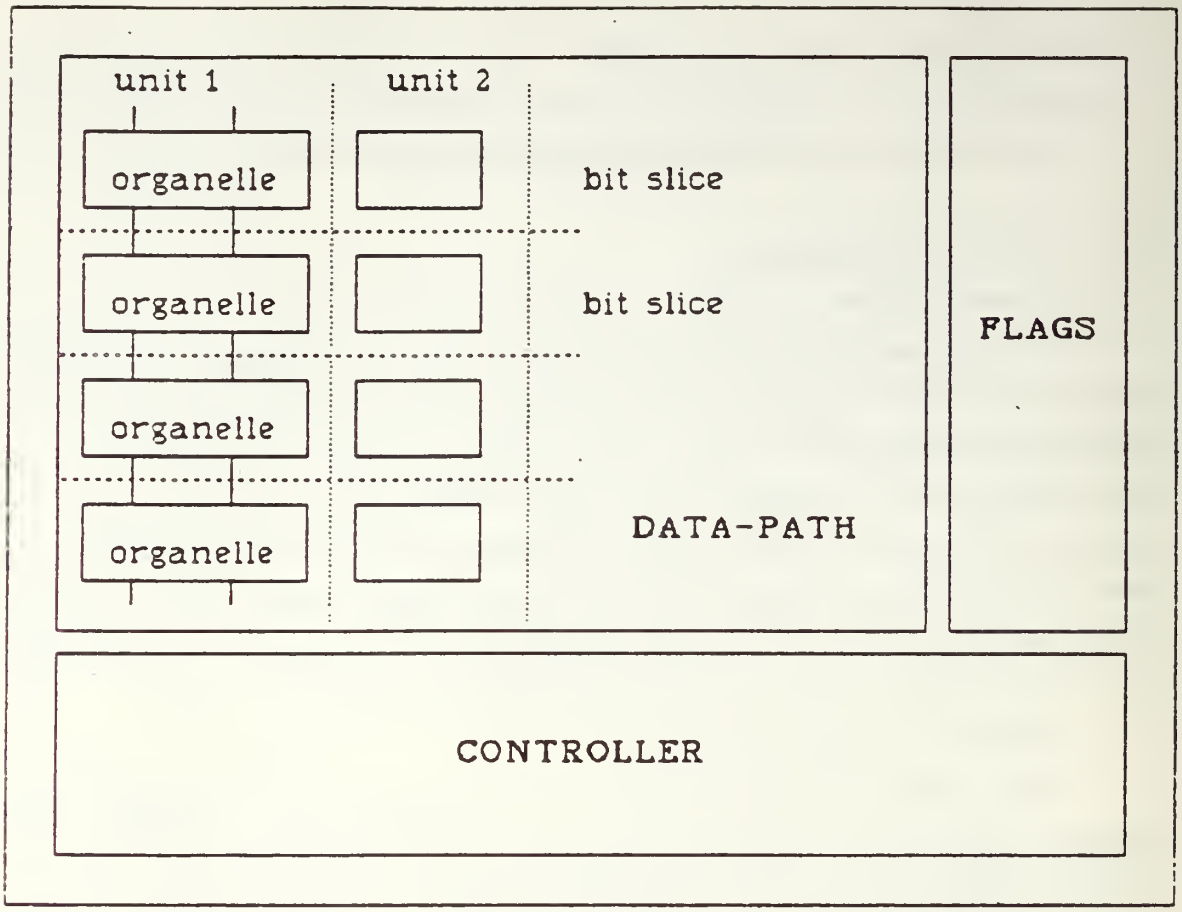

Figure 2.1 MacPitts Internal Layout.

5. Carries line up vertically to allow daisying of the connections between organelles.

Routing optimization was a determining factor in the placement of the inputs, outputs and daisies.

B. ORGANELLE DESIGN

1. Organelle Summary

Using the above guidelines, SCMOS organelles can be designed to perform the required functions [Ref. 9: p. 47]. Table 1 contains a list and a brief description of the functions of the arithmetic organelles. The SCMOS adder organelle was designed by A. Mullarky [Ref. 6: p. 41]. The organelles designed in this thesis are the three remaining arithmetic functions: the incrementor, decrementor and subtractor.

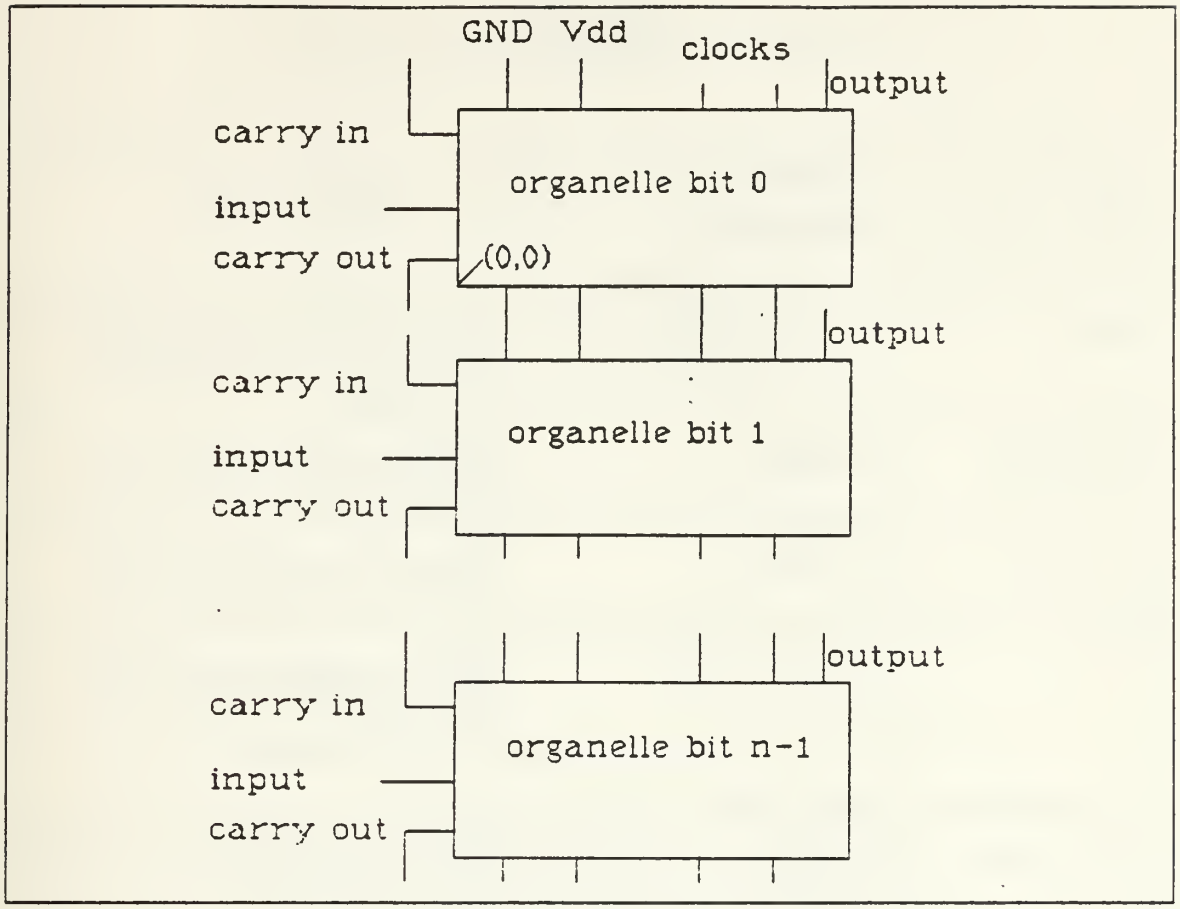

Figure 2.2 Organelle Layout.

2. **Design of an Organelle**

The design and test procedures will be presented for one organelle, the incrementor. The decrementor and subtractor were designed and tested in the same manner. The first step in the design process is to identify the required function of the incrementor [Ref. 9: p. 48].

> "(1+ integer) RETURNS: An integer which is the increment of the input integer." The incrementor is an arithmetic wordsize function where the integer is unsigned modulo 2.

Once the function is identified, a Karnaugh Map is used to derive the logical expressions of the incrementor, one expression for the p-structure and one for the n-structure of the CMOS technology. The incrementor is a combination of the AND and XOR functions. The truth table and resulting logical expressions are shown in

TABLE 1
MACPITTS ARITHMETIC FUNCTIONS

SCMOS ORGANELLE	LIBRARY FUNCTION	DESCRIPTION
Adder	(+ integer1 integer2)	Returns an integer which is the arithmetic sum of integer1 and integer2
Decrementor	(1- integer)	Returns an integer which is the decrement of the input integer
Incrementor	(1+ integer)	Returns an integer which is the increment of the input integer
Subtractor	(- integer1 integer2)	Returns an integer which is the arithmetic difference of integer1 and integer2 (i1 - i2)

Table 2 and Figure 2.3, respectively. The XOR designed by A. Mullarky was used for the XOR portion of the incrementor and decrementor organelles [Ref. 6: p. 100]. Figure 2.4 shows the corresponding CMOS transistor layout. From the transistor schematic, a stick diagram is generated to guide the layout of the organelle in the Magic (graphical layout editor) environment. The Magic designed organelle will have lambda = 1.5 (gen), generating CIF (Caltech Intermediate Format) code for the MOSIS (MOS Implementation System) SCMOS technology with a 3.0μm minimum feature size.

The design rules used to create the organelles include the Magic design rules [Ref. 10: Magic] and the design rules specified by A. Mullarky [Ref. 6: p. 29]. The following rules allow multiple organelles to abut without causing design rule violations:

- All I/O points are on first metal with inputs on the left edge of the organelle and outputs on the top edge.
- The transistors are oriented vertically (diffusion running vertically and polysilicon running horizontally), with the p-type toward Vdd and the n-type toward GND.
- Substrate contacts are connected by metal to supply rails, with one contact for every four or five transistors.

TABLE 2
TRUTH TABLE

IN	CIN	OUT	COUT
0	0	0	0
1	0	1	0
1	1	0	1
0	1	1	0

Karnaugh Map Generated Expressions:

OUT(p-structure) = (IN)(CIN') + (IN')(CIN)
OUT(n-structure) = (IN')(CIN') + (IN)(CIN)

COUT(p-structure) = (IN)(CIN)
COUT(n-structure) = (IN') + (CIN')

Note: The single quote (') represents the 'not' of the variable.

Figure 2.3 Logic Expressions.

- First metal and polysilicon are used for power and signal routing within organelles.
- External CLOCK, Vdd and GND connections are on second metal only, and run the full width of the organelle, perpendicular to the input. No other second metal is used in the organelles.
- All external connections to I/O, CLOCK, Vdd and GND end at least 5 units past all transistors.

- All external connections to I/O, CLOCK, Vdd and GND end at least 4 units past all substrate contacts.
- All external connections to I/O, CLOCK, Vdd and GND end at least 2 units past polysilicon.
- All external connections to I/O, CLOCK, Vdd and GND end at least 2 units past first metal that is not an I/O point.
- All external connections to I/O, CLOCK, Vdd and GND end at least 2 units past second metal that is not a CLOCK, Vdd or GND point.

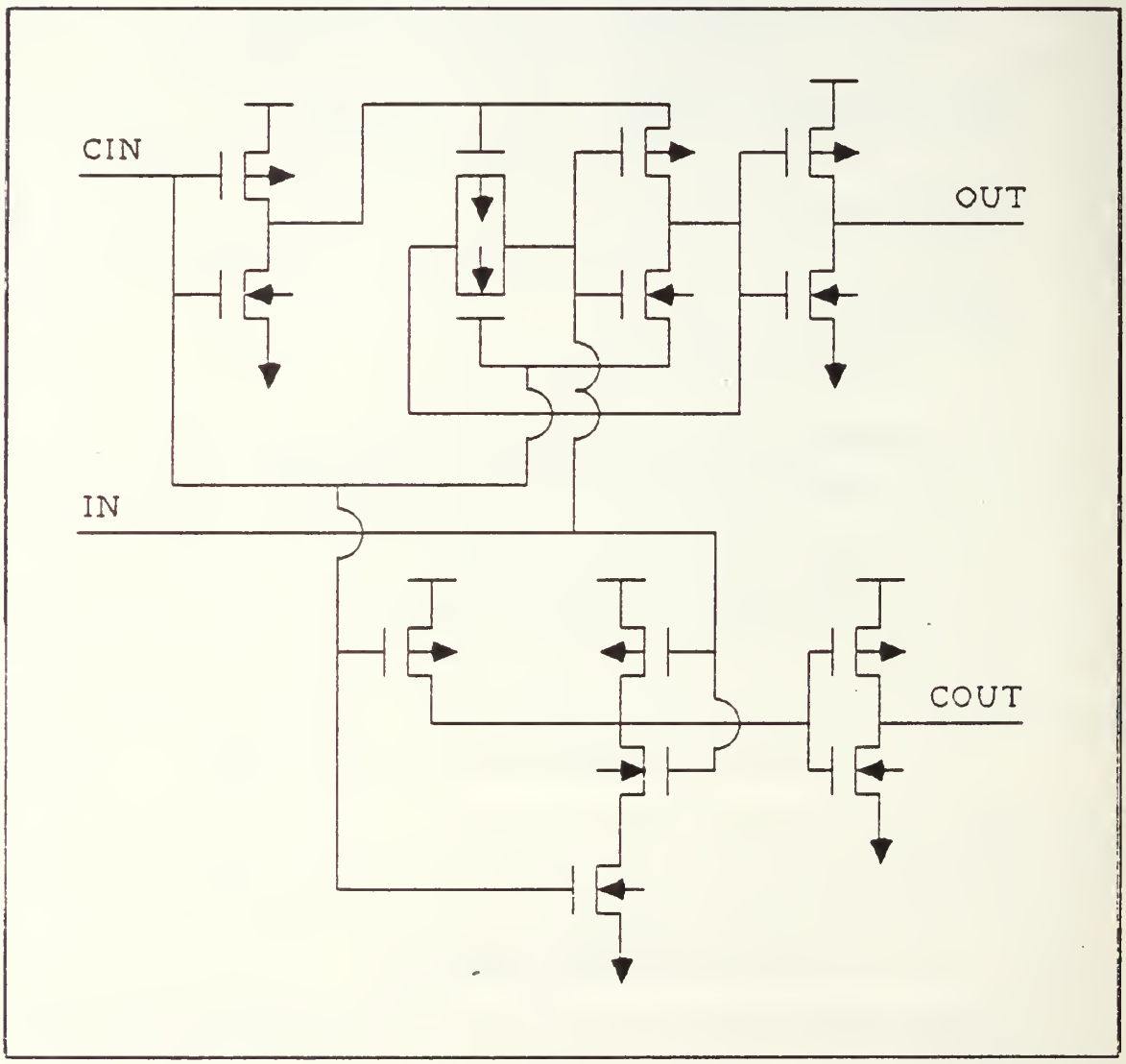

Figure 2.4 Transistor Schematic (Incrementor).

In addition to the above design rules, the SCMOS organelles were designed to have their longest dimension along the width (y-axis) of the data-path. The opposite is true for the NMOS organelles, which have their longest dimension along the length (y-axis) of the data-path. The reason for designing the SCMOS organelles in this way can be seen by examining Figure 2.5. The length of the data-path grows as the number of chip functions increase. The width depends upon the word length of the chip. For large computer chips, the number of functions is likely to be much greater than the number of bits in the word. In the NMOS layout, top of Figure 2.5, this results in the data-path growing much longer than it is wide. The controller does not grow in length proportional to the growth of the data-path. Therefore the area to the right of the controller is unused (white space in Figure 2.5). The longer the data-path, the more area to the right of the controller is wasted. In the SCMOS chip, the organelles are designed short and wide, to reduce the growth of the chip along the length of the data-path and increase the growth along the width. This results in less wasted chip area next to the controller as illustrated in the bottom half of Figure 2.5.

3. Logic Simulation

The completed Magic layout of the incrementor was tested using the switch-level simulator MOSSIM (MOS SIMulator). MOSSIM was used because of its capability of simulating transmission gates. Individual strength values can be assigned to the p-type and n-type pass transistors in the transmission gate. The strength is representative of the conductance of the transistor when it is turned on. Therefore, assigning a strength of '1' (lower conductance) to p-type and n-type pass transistors, and a strength of '2' (higher conductance) to the logic transistors forces the current through the transmission gate. This prevents generation of an undefined or 'X state'. [Ref. 11]

C. SPICE SIMULATION

The operation of the incrementor transistor layout was simulated using SPICE [Ref. 12]. A complete description of the SPICE simulation process is given in Appendix A. Table 3 shows the timing data generated for the incrementor.

The organelle was designed to obtain approximately equal rise and fall times. A wide variation in these times can be corrected by varying the width of the p-type transistors. The relationship of the carrier mobilities is shown in Equation 2.1.

$$\mu_n \sim 2\mu_p \tag{2.1}$$

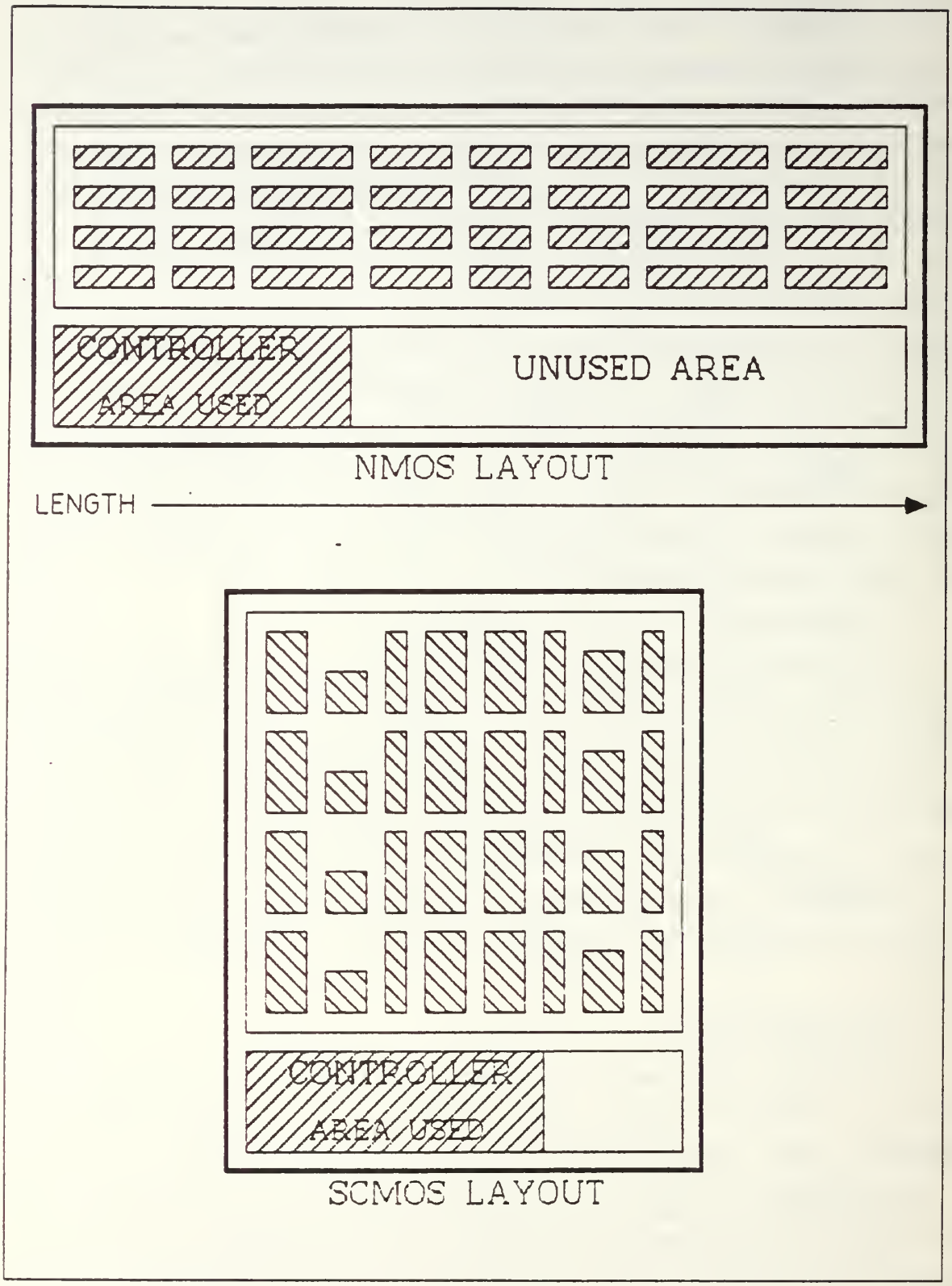

Figure 2.5 Data-path Dimensions.

TABLE 3
SPICE GENERATED TIMING DATA

PARAMETER	FANOUT	t_{dr}	t_{df}	t_r	t_f
IN → OUT	1	2.4ns	2.7ns	2.0ns	1.4ns
	4	3.2ns	3.3ns	3.8ns	2.1ns
IN → COUT	1	2.4ns	2.0ns	2.1ns	1.2ns
	4	3.3ns	2.4ns	3.7ns	1.8ns
CIN → OUT	1	1.7ns	2.7ns	1.6ns	1.3ns
	4	2.3ns	3.4ns	3.6ns	2.0ns
CIN → COUT	1	1.9ns	1.9ns	1.7ns	1.0ns
	4	2.7ns	2.3ns	3.4ns	1.7ns

Theoretically, to have the same rise and fall times, the channel width for the p-type transistors must be approximately two times the channel width of the n-type transistors. Equation 2.2 shows the relationship of the channel widths.

$$W_p \sim 2W_n \tag{2.2}$$

However, increasing the channel width of the p-type transistors also increases the layout area and the dynamic power dissipation of the organelle. [Ref. 13: p. 141]

The way that these transistors are combined also affects the rise and fall times. For a series connection of k (integer) p-type transistors or k n-type transistors, assuming the transistors are identical, the rise and fall times of the series combination will be kt_r and kt_f respectively (t_r and t_f are the rise and fall times of a single transistor). A parallel combination of these same transistors will result is the rise and fall times of the parallel combination to be t_r / k and t_f / k, assuming all the transistors are turned on at the same time. [Ref. 13: p. 181]

The determination to vary the transistor size from the minimum size of 3μm length and 4.5μm width, in an attempt to obtain equal rise and fall times, was based on the transistor layout of each organelle. This can been seen from the incrementor and decrementor transistor schematics. In the incrementor schematic, Figure 2.4, the transistor path to the carry output (COUT) is composed of two p-type transistors in parallel and two n-type transistors in series. Assume that one minimum size p-type transistor has a rise time of 4ns (four nanoseconds) and one minimum size n-type transistor has a fall time of 2ns ($\mu_n \sim 2\mu_p$). This combination will result in a rise time of 4ns / 2 = 2ns, and a fall time of 2 x 2ns = 4ns. Therefore, increasing the p-type transistor width in this case would make the rise and fall times of the output COUT even more unequal. A 1X drive design (p-type and n-type transistors of the same minimum size) was therefore used for the incrementor organelle.

The borrow output (BOUT) of the decrementor (Appendix B), is the result of a combination of two p-type transistors in series and two n-type transistors in parallel. Using the same rise and fall times used above for the minimum size transistor, the resulting rise and fall times would be 8ns for the p-type transistor combination and 1ns for the n-type transistor combination. In this case making the p-type transistor width equal to 2 times the n-type transistor width (2X drive design) would make the rise and fall times more equal. A 2X drive design was used for the decrementor organelle as well as the subtractor organelle.

D. ORGANELLES AND TEST RESULTS

The incrementor, decrementor and subtractor were designed and tested in the manner just described. Appendix B contains the schematic diagrams, truth tables, timing data and stipple plots of the Magic layouts.

Two Magic designs and their corresponding timing data are shown for the subtractor organelle. The SPICE generated timing data for the first design (Version 1), showed that the rise and fall times of the borrow out (BOUT) were significantly longer than the rise and fall times for the difference out (DOUT). This situation was improved in the second version by deleting two of the contacts in the output path of BOUT, and sharing the remaining contacts. This reduced the amount of capacitance in the BOUT output path, thereby decreasing the rise and fall times. Further reduction in the rise and fall times by sharing additional contacts was not possible due to the structured layout pattern of the transistors in the Magic layout.

III. ORGANELLE INSERTION INTO MACPITTS

A. THE MACPITTS ORGANELLE

The SCMOS organelles designed to perform the arithmetic functions described in Chapter II are now inserted into the MacPitts Silicon Compiler. Each SCMOS organelle will replace the NMOS version in the data-path. The method by which the MacPitts Silicon Compiler calls and lays out an organelle is discussed by E. Malagon [Ref. 7: p. 24]. The details of the operation will not be presented. However, the changes that must be made to the MacPitts code in order to insert an SCMOS organelle are better understood after a brief description of the pertinent MacPitts code.

1. Call and Layout of an Organelle

The two sections of MacPitts code that are accessed to layout an organelle in the data-path are the organelles.l file and the library file. The layout of each organelle is initiated by the compiler call of the function 'layout-gen-form'. This function passes an instantiate command to the organelle data structure contained in the library file. The organelle data structure in the library file includes the following items:

- (organelle <name> <# control-lines> <#parameters>
  <#testlines> result?
  (gen-form)
  (sim-form))

The gen-form describes such data as the length and width of the organelle, the input and output connection points, the number of transistors and the power requirements. MacPitts uses this information to place the organelle within the data path. This ensures that the organelles will be properly spaced and all connection lines generated by MacPitts will line up with the input and output points on the boundary of the SCMOS organelle. This gen-form data must be changed to match the characteristics of the SCMOS organelle.

The actual code which describes the geometric layout of the organelle is in the organelles.l file. This code is accessed by a call to layout the organelle (layout-incrementor-organelle or layout-1+-organelle) contained in the gen-form of the library file data structure.

2. Modifications Required to Insert an Organelle

Replacing an arithmetic NMOS organelle with a SCMOS organelle requires the modification of two MacPitts files. The gen-form entry describing the SCMOS organelle must be written and inserted into the library file, and the SCMOS organelle's L5 (Lincoln Laboratory's LISP based Layout Language) geometric layout code must be inserted into the organelles.l file.

B. MODIFYING THE LIBRARY FILE

The content of the library file gen-form is described in detail by E. Malagon [Ref. 7: p. 41]. The basic format of the gen-form is shown in Figure 3.1. For each organelle two versions (bit 0 and bit n) are inserted into the compiler. Bit 0 is used for the LSB (least significant bit) and bit n is used for the succeeding bits of a multiple bit layout. The bit 0 of each of the organelles has its carry in bit set to a predetermined value. For example, the incrementor has the carry-in input of the LSB wired to Vdd, thereby creating the increment to the first bit of the word which will then propagate down to succeeding bits.

The library entries must allow for variations in the connection points of the organelles as well as the size of the organelles. Bit n and bit 0 of the organelles may have different widths, or terminal connections at slightly different locations. Therefore, the LISP (LISt Processing) code was altered to include a conditional test for the LSB.

C. MODIFYING THE ORGANELLE FILE

The two versions of the organelle also require two entries in the organelles.l file. Since the SCMOS organelles are not built hierarchically, the two versions, bit 0 and bit n, must be created as separate organelles in the Magic environment. These entries will be selected based on the results of a conditional test for bit0, as in the library file. The code generated by the 'cif write' function on the ISI workstation, is a box description of the layers in the Magic designed organelle. MacPitts uses a rectangular description of the layout called the L5 version. Therefore, before insertion of the layout code into the organelles.l file, the code is converted from CIF (CalTech Intermediate Form) code to L5 code via the conversion routine 'cifdef.l' [Ref. 7: p. 110].

D. GENERATING THE MACPITTS LAYOUT

Once the library and organelles.l files have been changed and the new compiled version of organelles.l (organelles.o) created, the layout of the organelle within a

```
(lambda (info bit word-length drive ratio)
 (cond
  ((eq info 'instantiate)
     (first-quadrant (layout-'organelle-name'-organelle drive ratio)))
  ((eq info 'length) #)
  ((eq info 'width) #)
  ((eq info 'inputs) '(# #))
  ((eq info 'daisy) (#))
  ((eq info 'test) (#))
  ((eq info 'output-type) '(ratio))
  ((eq info 'drive) ())
  ((eq info 'conductivity) (quotient # #))
  ((eq info 'vdd) '(#))
  ((eq info 'gnd) '(#))
  ((eq info 'output) '(#))
  ((eq info 'number-transistors) '(n ()) ))
```

Figure 3.1 Gen-form Format.

MacPitts generated chip can be tested. This layout is created by the MacPitts input program.

1. **The Input Program**

The MacPitts input program, the .mac file, is written to generate the particular organelle function. A sample .mac program is shown in Figure 3.2. The .mac program tells MacPitts what function the circuit is to perform, the word-length of the data-path, and the number and type of ports (Vdd, GND, input, output and internal).

2. **Running MacPitts**

After the .mac program is written, the chip is generated by entering the MacPitts environment and executing the .mac program as shown below:

- (macpitts < .mac filename > < options >)

```
(program incrementor 1
 (def 1 ground)
 (def a port input (2))
 (def result port output (3))
 (def carry port internal)
 (def 4 phia)
 (def 5 phib)
 (def 6 phic)
 (def 7 power)
 (always
  (setq result (1+ a))))
```

Figure 3.2 MacPitts Input Program (incrementor.mac).

The layout is then plotted using the Cifplot command on the CS-VAX computer. Windowing may be used to plot only the organelle portion of the chip so that the interconnections can be checked. Appendix D contains a complete description of the insertion process. A step by step procedure is given for conversion of the CIF code to L5 code; editing of the organelles.l file; making the organelles.o file; changing the library file entry; and macPitts generation and plotting of the inserted SCMOS organelle.

E. COMPARISON OF SCMOS AND NMOS LAYOUTS

1. Organelles

The NMOS and SCMOS organelles are compared in Table 4. The number of transistors shown is the total number in a single organelle. The size comparison was made from the MacPitts chip layouts, by measuring the area of the chip occupied by the organelle and its input and output routing connections. The number listed for size in Table 4 is the number of square microns within the inside boundary of the GND and Vdd rails, as shown in Figures 3.3a and 3.3b. As expected, even though the number of transistors in the SCMOS organelles is greater than or equal to the number of transistors in the NMOS organelles, the SCMOS organelles occupy a smaller area of the chip. The main reason for this is that the NMOS organelles were designed

TABLE 4
ORGANELLE COMPARISONS

	# TRANSISTORS		SIZE(sq. microns)	
	NMOS	SCMOS	NMOS	SCMOS
INCREMENTOR	10	14	21,084	18,480
DECREMENTOR	12	12	24,969	19,557
SUBTRACTOR	20	30	46,354	38,106

hierarchically, using fixed size submodules. The SCMOS organelles are custom designed, resulting in a more economic use of the chip area. The chip area occupied by the SCMOS organelles will decrease as routing optimizations are made. This is because the circuitous routing to the inputs and outputs is inside the GND and Vdd rail boundaries used for the area measurements.

2. Chip Layouts

The MacPitts chip layouts generated in verifying the SCMOS organelle insertions are shown in Appendix C. A one-bit and four-bit layout are shown for each of the organelles designed. The same MacPitts generation using the NMOS organelles is also shown. Note that although the SCMOS version of MacPitts contains the modification of the chip to two clocks [Ref. 7: p. 88], the three clock version was created so that the NMOS and SCMOS chips would have the same number of ports and the layouts could be compared visually. The MacPitts generations of the SCMOS chips were completed using the 3 μm minimum feature size (3 micron) process (hybrid option of MacPitts execution statement). Although the 3 micron process is not supported in the NMOS technology, the NMOS chips were also generated using the 3 micron process. This was done for size comparison purposes, so that the NMOS stipple plots shown in Appendix C could be made in the same scale as the SCMOS plots.

The taxi meter chip is used to illustrate the changing dimensions of the data-path as the compiler is converted from NMOS to SCMOS technology. Prior to this

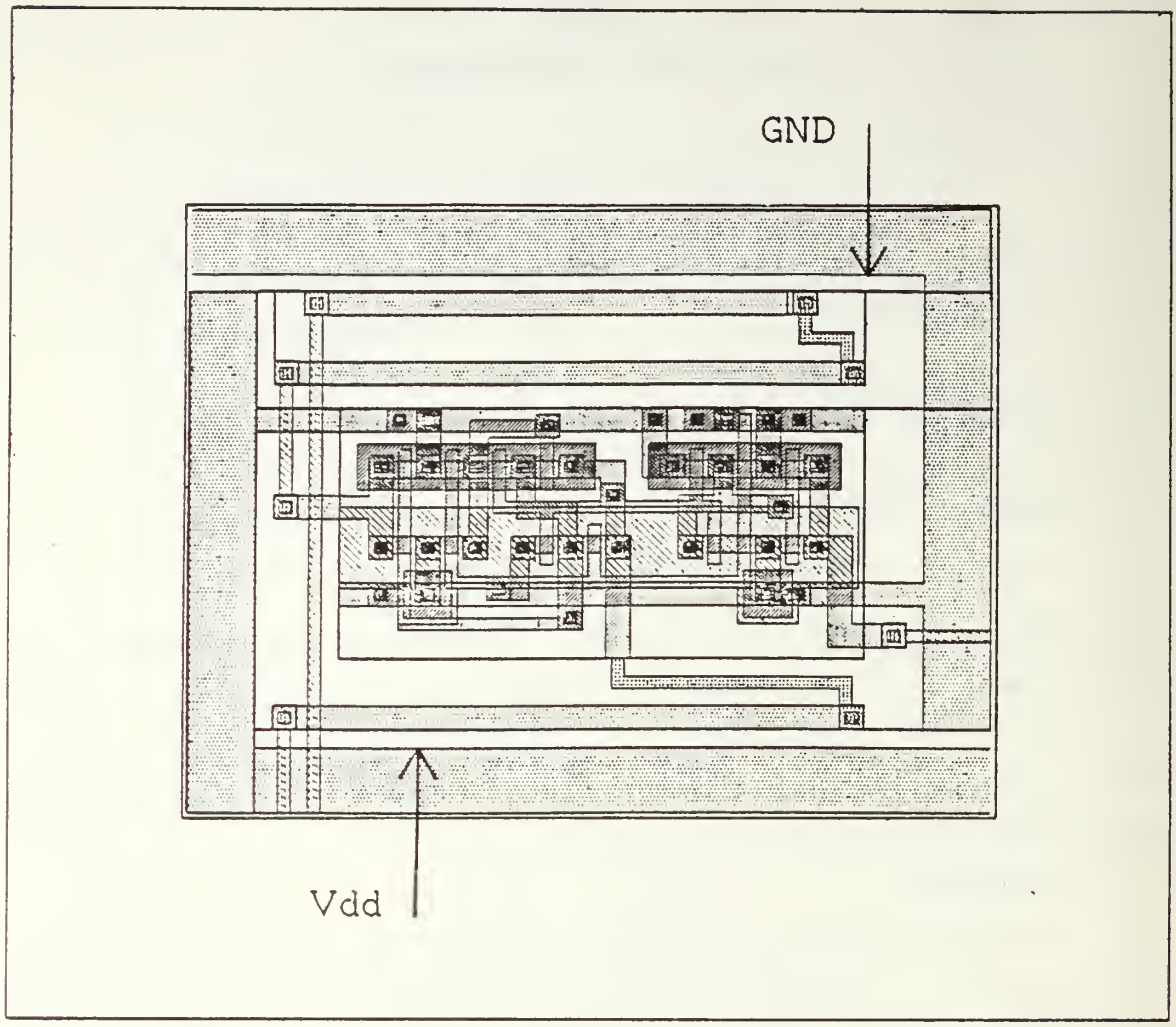

Figure 3.3a SCMOS Organelle Size Measurement.

thesis, there were two NMOS organelles remaining in the SCMOS taximeter chip layout. One of these (the incrementor) was designed in this thesis. Therefore, the NMOS taximeter chip and the updated SCMOS taxi meter chip are shown in Figure 3.4. [Ref. 7: p. 89]

A comparison of the two taxi meter chips reveals that although the wasted area to the right of the controller portion of the chip has decreased in the SCMOS version, the overall area of the chip is slightly larger than the NMOS chip:

- SCMOS chip dimensions - 2286 µm X 3086 µm
- NMOS chip dimensions - 3114 µm X 2200 µm

This is due in part to the existence of the one remaining NMOS organelle (multiplexor) in the SCMOS chip. Another factor increasing the area of the chip is circuitous input

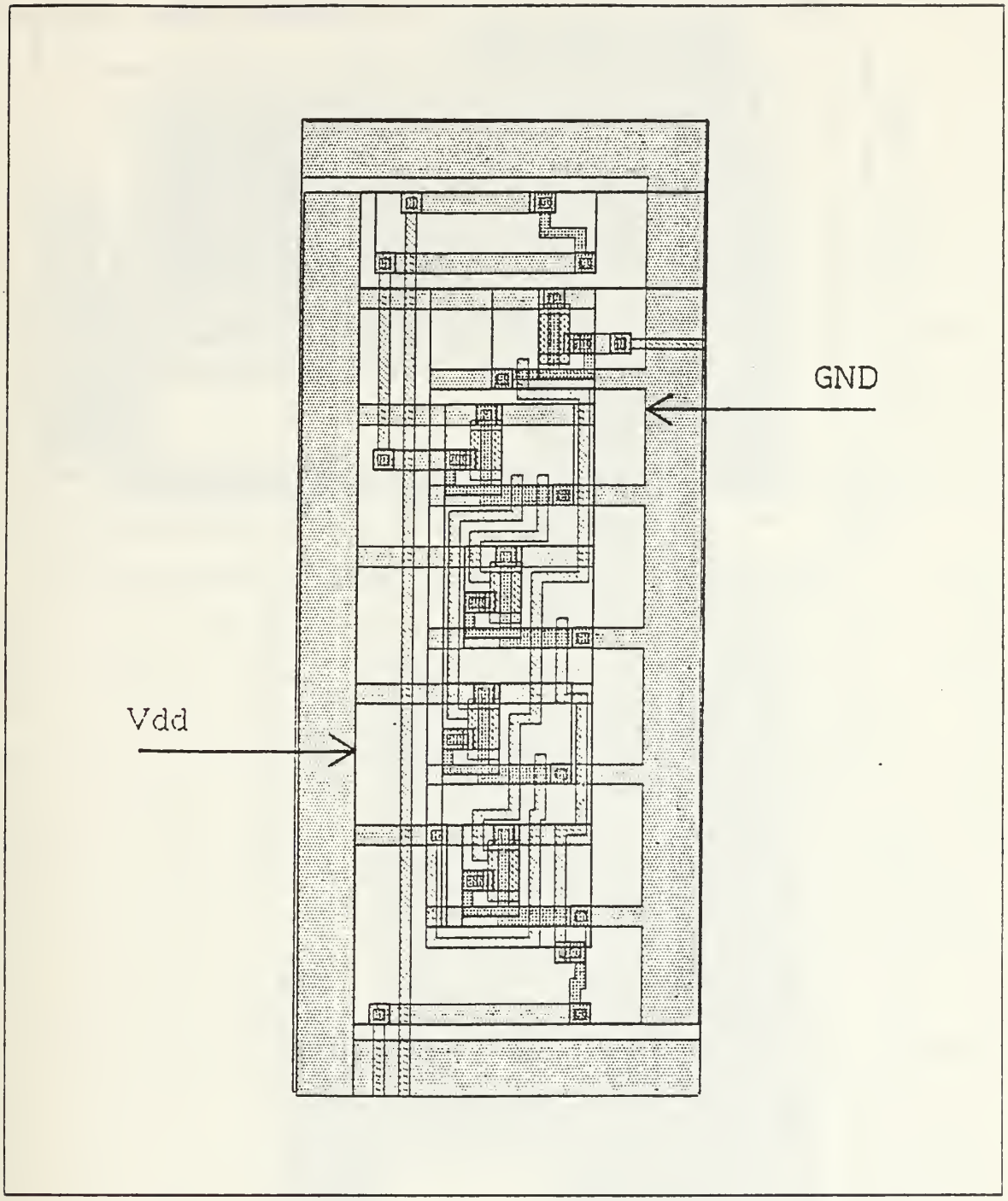

Figure 3.3b NMOS Organelle Size Measurement.

and output connections to each organelle. Once the planned routing optimizations are made, the data-path width will decrease sufficiently to allow the SCMOS taxi meter chip to be smaller than the NMOS taxi meter chip.

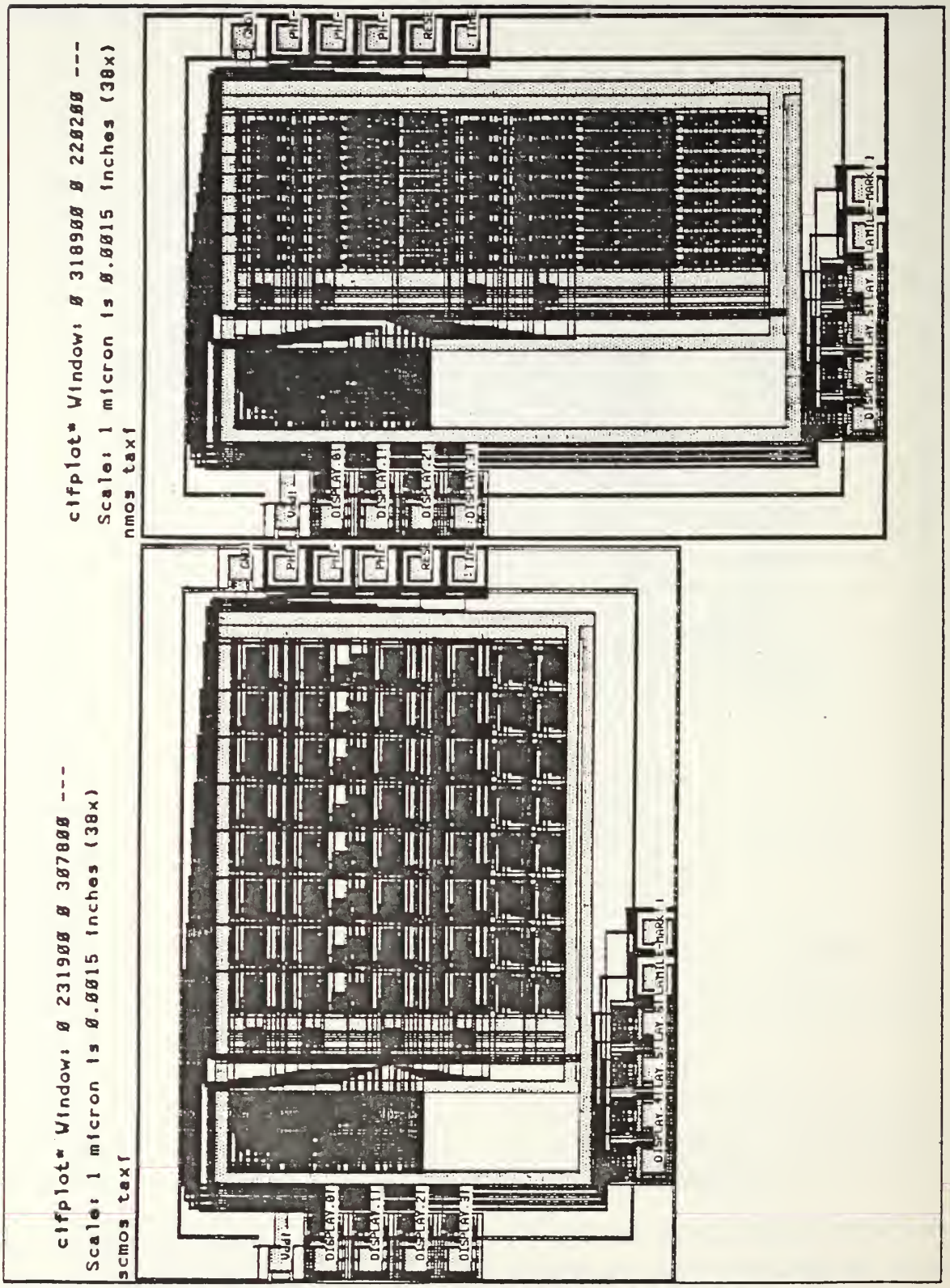
clfplot* Window: 0 231900 0 307800 ---
Scale: 1 micron is 0.0015 inches (38x)
scmos taxf
clfplot* Window: 0 318900 0 220200 ---
Scale: 1 micron is 0.0015 inches (38x)
nmos taxf

Figure 3.4 NMOS and SCMOS Taxi Meter Chips.

IV. THE MACPITTS ISI ENVIRONMENT

A. STATUS OF THE ENVIRONMENT

The completed arithmetic SCMOS organelles were inserted into the MacPitts environment on the ISI workstations. MacPitts was installed on the ISI's by J. Harmon [Ref. 8]. He made modifications to enhance the capabilities of the compiler on the ISI's, allowing use of Magic to view the completed MacPitts layouts. He also added processing options to the MacPitts command statement for the use of NMOS, SCMOS and hybrid processes, and their corresponding default minimum feature sizes. E. Malagon's work of inserting organelles was completed on the CS-VAX computer system [Ref. 7].

The first step in the insertion of the incrementor, decrementor and subtractor organelles was to enter the SCMOS organelles existing in the CS-VAX MacPitts code into the ISI MacPitts code. This involved the modification of three files: data-path.l, library and organelles.l. The SCMOS data-path.l file modified by E. Malagon was copied to the ISI MacPitts environment. The changes made to the NMOS data-path.l file by J. Harmon were edited into the SCMOS version. The new data-path.o (compiled version of data-path.l) was then created. In order to be able to load the updated version of data-path.o into MacPitts for testing, code was added to the library file to allow MacPitts to search the home directory for this file and load it at compile time. The code added is identical to the code inserted by J. Harmon for booting in the organelles.o file at compile time.

For the library and organelles.l files, the NMOS versions (on ISI workstations) were used and the SCMOS organelle insertions made by E. Malagon were reinserted into the NMOS file, creating the new SCMOS (hybrid) version. Code changes made by J. Harmon were then applied to the portions of SCMOS code that was reinserted. The changes made to these files are commented at the beginning of each file and at the site of the changes within the code. The MacPitts master files have not been changed pending the insertion of additional SCMOS organelles. Table 5 lists the three files and the changes made.

TABLE 5

ALTERED MACPITTS FILES

FILE	CHANGES MADE
data-path.l	- used SCMOS version created by E. Malagon and edited in the changes to the code identified by J. Harmon at the top of the NMOS ISI version of data-path.l
library	- used NMOS ISI version of library and edited in the organelle library entries inserted by E. Malagon, on the CS-VAX, making changes to these entries with respect to # symbols and in-line lambda entries noted by J. Harmon in the NMOS code - added SCMOS library entries for the incrementor, decrementor and subtractor
organelles.l	- used the NMOS ISI version of organelles.l and edited in the SCMOS organelle code entered by E. Malagon on the CS-VAX - added SCMOS organelle code for incrementor, decrementor and subtractor

B. THE CURRENT MACPITTS ENVIRONMENT

A list of the files residing in the working directories used to complete this thesis is given in Table 6. This is provided as an aide for future organelle entries. Changes may be made to a copy of these files, so that all SCMOS organelle entries will reside in one set of files. The goal of the ISI MacPitts environment is to have a separate set of files (library, organelles.l/.o, data-path.l/.o, etc.) for the NMOS and SCMOS versions of the compiler. The correct version can then be called by options specified in the MacPitts generation command. The notations of changes to be made to the SCMOS version of the library file by E. Malagon still remain in the code for future work.

TABLE 6

FILE DIRECTORY

Contents of ISI directory 'a/work/baumsta':

cifdef.l	conversion routine (CIF to L5)
dec2x.mag	decrementor bit n
dec2x_0.mag	decrementor bit 0
defs	SPICE definitions file
inc2x.mag	incrementor bit n
inc1x_0.mag	incrementor bit 0
sub2x.mag	subtractor bit n
sub2x_0.mag	subtractor bit 0
sub2x_Version1.mag	subtractor Version 1
subtractoropt2.mag	subtractor bit n extra copy

Contents of ISI directory 'a/work/baumsta/macpitts':

cifdef.l	conversion routine (CIF to L5)
data-pathscmos.l	updated SCMOS version
data-pathscmos.o	updated SCMOS version
data-path_old.o	NMOS version
dec.mac	1-bit MacPitts input program for decrementor
dec.obj	object file from MacPitts run of dec.mac
dec2x.L5	decrementor bit n L5 code
dec2x.cif	decrementor bit n CIF code
dec2x_0.L5	decrementor bit 0 L5 code
dec2x_0.cif	decrementor bit 0 CIF code
decnmos2.cif	MacPitts generated 1-bit NMOS decrementor chip
decnmos4.cif	MacPitts generated 4-bit NMOS decrementor chip
decrementor.mac	4-bit MacPitts input program for decrementor
decrementor.obj	object file from MacPitts run of decrementor.mac
decscmos2.cif	MacPitts generated 1-bit SCMOS decrementor chip
decscmos4.cif	MacPitts generated 4-bit SCMOS decrementor chip
inc.mac	1-bit MacPitts input program for incrementor
inc.obj	object file from MacPitts run of inc.mac
inc1x.L5	incrementor bit n L5 code
inc1x.cif	incrementor bit n CIF code
inc1x_0.L5	incrementor bit 0 L5 code
inc1x_0.cif	incrementor bit 0 CIF code
incnmos1.cif	MacPitts generated 1-bit NMOS incrementor chip
incnmos4.cif	MacPitts generated 4-bit NMOS incrementor chip
incrementor.mac	4-bit MacPitts input program for incrementor
incrementor.obj	object file from MacPitts run of incrementor.mac
incscmos1.cif	MacPitts generated 1-bit SCMOS incrementor chip
incscmos4.cif	MacPitts generated 4-bit SCMOS incrementor chip
libchanges	changes made to library file by J. Baumstarck
libeva	E. Malagon's version of library from CS-VAX
libjim3Sep	J. Harmon's version of library (3 Sep. visit)
library	holds NMOS and SCMOS version for testing
library_old	old version of library from 'macpitts' directory set up by J. Harmon
libscmos	SCMOS version of library with incrementor, decrementor and subtractor
libscmos2	backup copy of libscmos
organelles_old.o	old compiled version from 'macpitts' directory set up by J. Harmon
orgeva.l	E. Malagon's version of organelles.l from CS-VAX
orgjim3Sep	J. Harmon's version of organelles.l (3 Sep. visit)
orgscmos.l	SCMOS version of organelles.l with incrementor, decrementor and subtractor
orgscmos.o	compiled version of orgscmos
orgscmos2.l	backup copy of orgscmos.l

TABLE 6

FILE DIRECTORY (CONT'D.)

patterns	E. Malagon's patterns file from CS-VAX
sub.mac	1-bit MacPitts input program for subtractor
sub.obj	object file from MacPitts run of sub.mac
sub2x.L5	subtractor bit n L5 code
sub2x.cif	subtractor bit n CIF code
sub2x_0.L5	subtractor bit 0 L5 code
sub2x_0.cif	subtractor bit 0 CIF code
subnmos1.cif	MacPitts generated 1-bit NMOS subtractor chip
subnmos4.cif	MacPitts generated 4-bit NMOS subtractor chip
subscmos1.cif	MacPitts generated 1-bit SCMOS subtractor chip
subscmos4.cif	MacPitts generated 4-bit SCMOS subtractor chip
subtractor.mac	4-bit MacPitts input program for subtractor
subtractor.obj	object file from MacPitts run of subtractor.mac
taxi.mac	MacPitts input program for taxi meter chip
taxinmos.cif	MacPitts generated NMOS taxi meter chip
taxinmos.obj	object file from MacPitts run of taxi.mac
taxiscmos.cif	MacPitts generated SCMOS taxi meter chip
taxiscmos.obj	object file from MacPitts run of taxi.mac

Contents of CS-VAX directory 'nps-cs.arpa-/work/baumsta':

dec2x.mag	backup copy of decrementor bit n
dec2x_0.mag	backup copy of decrementor bit 0
inc1x.mag	backup copy of incrementor bit n
inc1x_0.mag	backup copy of incrementor bit 0
model.spice	SPICE model parameters
modelw.spice	SPICE worst case model parameters
sub2x.mag	backup copy of subtractor bit n
sub2x_0.mag	backup copy of subtractor bit 0
*.cif	miscellaneous CIF files for plotting

Contents of CS-VAX directory 'nps-cs.arpa-/work/baumsta/Mac':

libscmos.l	backup copy of SCMOS organelles.l file
patterns	backup copy of E. Malagon's patterns file
organelle_old.o	backup copy of J. Harmon's version
organelles_old.l	backup copy of J. Harmon's version
MacPitts files	remaining files copied from E. Malagon's files on the CS-VAX

V. CONCLUSION

A. REVIEW OF THESIS WORK

1. Scope of Thesis

This thesis was the first to include both the design and insertion of SCMOS organelles. Although the existing documentation, the completed theses, provided sufficient information to accomplish both tasks, the scope of the thesis had to be scaled down to a workable level; the design and insertion of three arithmetic organelles. The design and insertion process was completed for one organelle and the process documented before the remaining organelles were completed. The first design and insertion therefore took a disproportionate amount of time.

2. Long Learning Curve

The knowledge gained in completing the first organelle significantly decreased the time needed for the remaining organelles. However, this knowledge cannot be passed directly to new thesis students tasked with designing and inserting the remaining organelles. An attempt was made to shorten this learning curve, by writing step-by-step procedures for the SPICE simulation and MacPitts insertion processes, and including these procedures as Appendices. This will aide the thesis student involved in creating SCMOS organelles for the data-path, but as the varying types of organelles are placed in different portions of the floor plan and by different MacPitts programs, there is a certain amount of knowledge the designer must gain as the first insertion is attempted. This is compounded by the fact that some understanding of the MacPitts code is required before the design phase in order to create optimum designs that will fit into the floor plan of MacPitts.

B. STATUS OF THE MACPITTS SILICON COMPILER

1. SCMOS Organelles

A summary of the organelles that remain to be designed is given by E. Malagon. The list is not repeated here, as the only changes to that list are the completion of the incrementor, decrementor and subtractor (the '1+', '1-' and 'subtract' functions). [Ref. 7: p. 95]

2. Conversion Efforts

The design and insertion of the SCMOS organelles is just a portion of the entire MacPitts Silicon Compiler conversion process. At the completion of the theses currently in process, there will still be a significant amount of work to complete the conversion of MacPitts. There is a need for a single thesis student to combine the changes made by the concurrent theses and to update the MacPitts master files, so that the cumulative effect of the theses can be realized. Assigning the responsibility of updating the MacPitts master files to individual thesis students after completion of each conversion effort, is unrealistic. The amount of knowledge required of the MacPitts code and the coordination required to allow multiple updates within a single quarter is prohibitive.

The list of conversion requirements grows as new problems are identified during the conversion process. Although commercial silicon compilers are available, and would be useful for thesis design projects, the knowledge gained from getting into the heart of a silicon compiler and understanding how the pieces fit together is something that cannot be provided in the classroom environment. The use of a commercial silicon compiler to design VLSI chips cannot provide the thesis student with the opportunity to learn hands-on, how a silicon compiler is designed: the floor plan, data-path architecture, hierarchical or custom module design, the level of designer interface, and the corresponding tradeoffs. The conversion of the MacPitts Silicon Compiler from NMOS to SCMOS technology remains a viable thesis source if this knowledge gain is recognized.

APPENDIX A
SPICE SIMULATION

This appendix describes the SPICE simulation of a CMOS organelle designed in the Magic environment on an ISI workstation. The SPICE simulation is also run on the ISI workstation. Reference is made throughout the text to SPICE "deck" and "cards". Each card represents a statement in the SPICE command file referred to as the SPICE deck. This terminology was used to be consistent with the SPICE User's Manual [Ref. 12]. For <filename> use the organelle filename.

1. Create the SPICE deck:
   a. Enter Magic, loading the file.
      - %magic <filename>
   b. Extract the organelle and quit Magic:
      - >:extract
      - >:quit
   c. Create <filename>.sim:
      - %ext2sim <filename>
   d. Use the vi editor to create the file named "defs" and enter the following definition statements:
      - def n CMOSN
      - def p CMOSP
   e. Create the SPICE file transistor cards:
      - %sim2spice -d defs <filename>.sim

      This creates a file called <filename>.spice containing the transistor cards and the capacitor cards. All the other cards of the SPICE deck will have to be input manually. Note that sim2spice ignores all the resistance values. Therefore, disregard the error messages "(line #) format error: R".
   f. From this deck a transistor schematic can be generated for comparison to the original schematic used to create the Magic design. Before printing a copy of the deck using the "%print <filename>.spice" command, delete the characters on the first line of the deck after the comment "***SPICE DECK created from <filename>.sim, tech=scmos". Using the "%lpr <filename>.spice" command does not require deletion of the characters. To simplify the process of reconstructing the schematic from the SPICE node numbers, refer to a stipple or Magic plot. The numbering of the transistors depends on the orientation of the cell when it was extracted. The transistors are numbered from left to right and top to bottom.

2. For realistic timing data, simulate a load and drive using the simulation model described by Mullarky [Ref. 6: p. 18]. The simulation model used for the incrementor organelle is shown in Figure A.1. Replace the inverters on the input and output with the einv1x version. Use this for a fanout of one, and for a fanout of four, replace the inverters on the output by the einv4x version.

   a. Each input should be buffered by two inverters and each output should have a load of one inverter (einv1x or einv4x).

   b. Use the design rule checker and MOSSIM to confirm correct operation of the altered organelle. Make these changes in a file other than the one containing the original design.

3. Once the test organelle is ready, obtain a stipple plot and annotate the node numbers on the plot as shown in Figure A.2. Repeat the SPICE generation procedure using the test organelle:

   - > :extract (in Magic)
   - %ext2sim <testfile> (in UNIX window)
   - %sim2spice -d defs <testfile>.sim

4. Create a file called model.spice to hold the ".MODEL" cards [Ref. 12: p.13]. Enter the .MODEL cards using the MOSIS NMOS and PMOS (P-type MOS) transistor parameters as listed in Table 7 [Ref. 6: p. 13]. If using the worst case parameter values, the ".TEMP" card must be used [Ref. 12: p. 19]. Concatenate the model file to the end of the <testfile>.spice file:

   - %cat <testfile>.spice model.spice > temp.spice

   This puts the concatenated version into the file temp.spice. It is better to use a more descriptive name than temp, and add a suffix for fanout. For example, use <filename>1.spice for a fanout of one and <filename>4.spice for a fanout of four.

5. Add the remaining SPICE deck cards, using the stipple plot, Figure A.2, as a guide.

   a. Add between the comment line and the first transistor card at the top of the file, the "VDD", "VPS" and "VNS" cards:

      - VDD Vdd node# GND node# DC value
        (+ node) (- node) (5)
      - VPS P-substrate node# GND node# DC value
        (+ node) (- node) (5)
      - VNS n-substrate node# GND node# DC value
        (+ node) (- node) (0)

      Be sure to use node number 0 as the GND node number. This may have to be changed in the other cards of the SPICE file. All GND connections should be node zero.

b. Add "VIN" cards for each input of the circuit. One of the inputs should be pulsed and the rest set to constant values. This enables measurement of the rise and fall times for each output based on a particular input. Place these cards after the transistor cards and before the capacitor cards. For the constant inputs:

- VIN# input node# GND node# DC value
  (+ node) (- node) (0 or 5)

For the pulsed inputs:

- VIN# input node# GND node# PULSE(....)
  (+ node) (- node)

The pulse fields (V1 V2 TD TR TF PW PER) are described below:

- V1 - initial value (volts)
- V2 - pulsed value (volts)
- TD - time delay (seconds)
- TR - rise time of pulse (seconds)
- TF - fall time of pulse (seconds)
- PW - pulse width (seconds)
- PER - period of the pulse = TD + TR + TF + PW (seconds)

Note, the pulse width and delay time can be lengthened to allow the outputs to settle before the inputs are changed. [Ref. 12: p.8]

6. The ".TRAN" and ".PLOT" cards are placed after the capacitor entries and before the ".MODEL" cards.

   a. The ".TRAN" card describes the plotting of the waveforms:

   - .TRAN TSTEP TSTOP

   These are the minimum inputs required. TSTEP is the plotting increment used for the printer and TSTOP is the final plot time. TSTART may also be specified (0 is the default). [Ref. 12: p. 25]

   b. The ".PLOT" card is used to specify the numbers of the nodes whose voltages are to be plotted and the range of plotted values:

   - .PLOT TRAN V(node#) V(node#) ... V(node#) (voltage range)
     (0,5)

   Note that the node numbers of the varied inputs are different from the actual input node numbers of the circuit under simulation. [Ref. 12: p. 27]

7. The ".END" card is required at the end of the SPICE deck. A sample SPICE deck is shown in Figure A.3.

8. Run the SPICE simulation by entering:

   - %spice <filename> 1.spice results

When the SPICE run has completed, print the results file. Any filename can be used for the "results" file in the command line.

9. One SPICE run will have to be done to allow the measurement of the rise and fall time of each output corresponding to the transition of each input waveform. For example, to generate the timing waveforms for the incrementor organelle which has two inputs (IN and CIN) and two outputs (OUT and COUT), a SPICE run should be made for each of the following conditions:

IN to OUT (pulse IN and set CIN = 0)
IN to COUT (pulse IN and set CIN = 1)
CIN to OUT (pulse CIN and set IN = 0)
CIN to COUT (pulse CIN and set IN = 1)

10. To obtain rise and fall times use the definitions specified by Mullarky [Ref. 6: p. 40]:

- Rise time (t_r) - the time for the output waveform to rise from 10% to 90% of its final value.

- Fall time (t_f) - the time for the output waveform to fall from 90% to 10% of its final value.

- Delay rise time (t_{dr}) - the time difference between the input waveforms 50% level and the corresponding output waveforms 50% rise level.

- Delay fall time (t_{df}) - the time difference between the input waveforms 50% level and the corresponding output waveforms 50% fall level.

The rise and fall times are also discussed by Weste and Eshraghian [Ref. 13: p. 137].

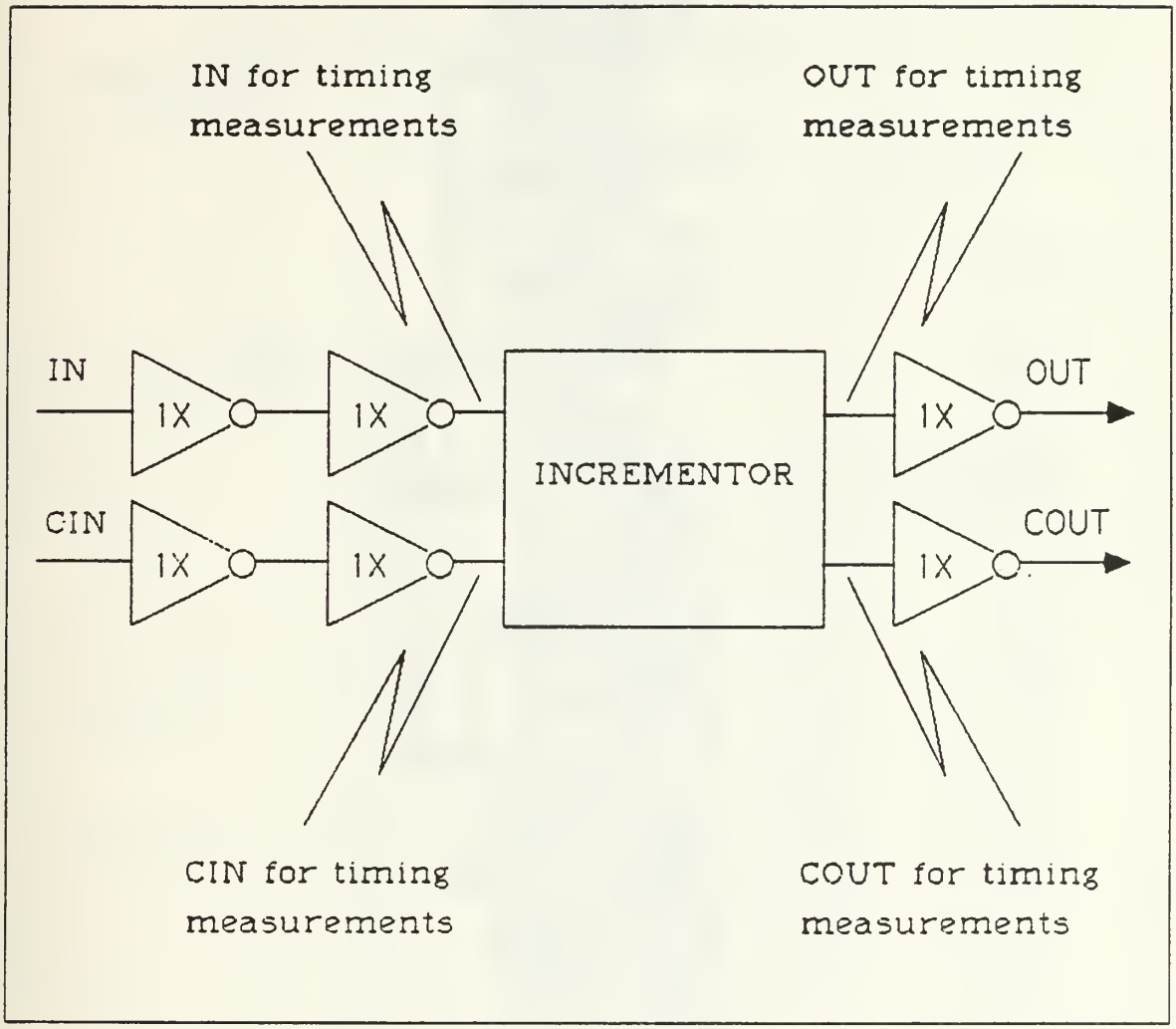

Figure A.1 SPICE Simulation Model.

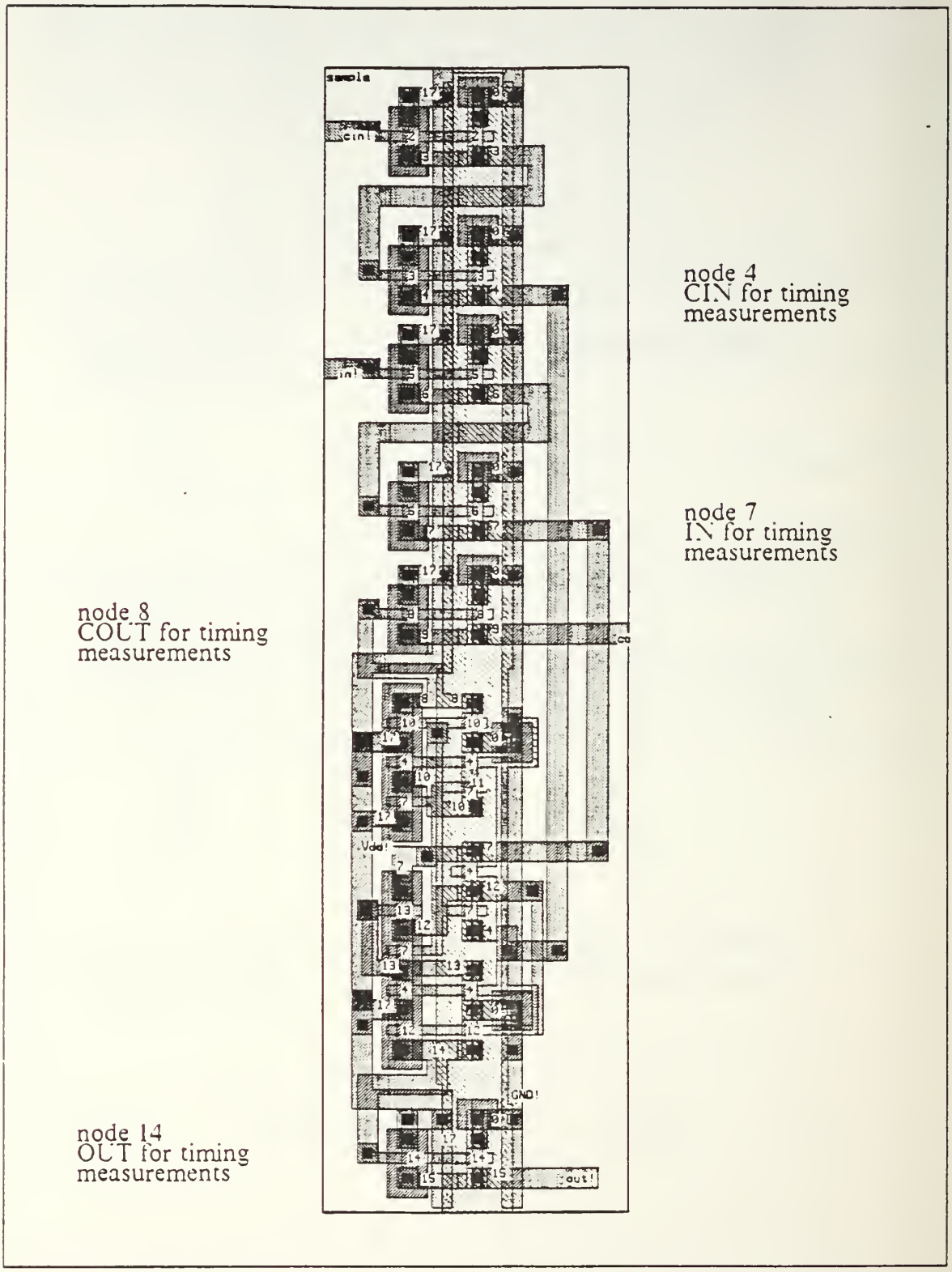

Figure A.2 Test Organelle Stipple Plot With Node Numbers.

TABLE 7

MOSIS TRANSISTOR PARAMETERS

TYPE	NOMINAL		WORST CASE	
	NMOS	PMOS	NMOS	PMOS
LEVEL	2.000	2.000	2.000	2.000
VTO (V)	0.827	-0.895	0.909	-0.984
KP (A/V)	3.29d-05	1.53d-05	3.29d-05	1.53d-05
GAMMA (V**0.5)	1.360	0.879	1.360	0.879
PHI (V)	0.600	0.600	0.600	0.600
LAMBDA (1/V)	1.60d-02	4.71d-02	1.60d-02	4.71d-02
CGSO (F/m)	5.20d-10	4.00d-10	5.20d-10	4.00d-10
CGDO (F/m)	5.20d-10	4.00d-10	5.20d-10	4.00d-10
RSH (ohm/sq.)	25.000	95.000	25.000	95.000
CJ (F/m**2)	3.20d-04	2.00d-04	3.20d-04	2.00d-04
MJ	0.500	0.500	0.500	0.500
CJSW (F/m)	9.00d-10	4.50d-10	9.00d-10	4.50d-10
MJSW	0.330	0.330	0.330	0.330
TOX (m)	5.00d-08	5.00d-08	5.00d-08	5.00d-08
NSUB (1/cm**3)	1.00d+16	1.2d+14	1.00d+16	1.12d+14
NSS (1/cm**2)	0.0d+00	0.0d+00	0.0d+00	0.0d+00
NFS (1/cm**2)	1.23d+12	8.79d+11	1.23d+12	8.79d+11
TPG	1.000	-1.000	1.000	-1.000
XJ (m)	4.00d-07	4.00d-07	4.00d-07	4.00d-07
LD (m)	2.80d-07	2.80d-07	2.80d-07	2.80d-07
UO (cm**2/V-s)	200.000	100.000	130.000	65.000
UCRIT (V/cm)	9.99d+05	1.64d+04	9.99d+05	1.64d+04
UEXP	0.001	0.153	0.001	0.153
VMAX (m/s)	1.00d+05	1.00d+05	1.00d+05	1.00d+05
NEFF	0.010	0.010	0.010	0.010
DELTA	1.241	1.938	1.241	1.938
TEMP (C)	27.00	27.00	125.00	125.00
POWER (V)	0.00	5.00	0.00	4.50

```
SPICE DECK created from inc1x.sim, tech=scmos
VDD 17 0 5
VPS 16 0 5
VNS 1 0 0
M1 3 2 0 1 CMOSN L=3.0U W=4.5U
M2 4 3 0 1 CMOSN L=3.0U W=4.5U
M3 6 5 0 1 CMOSN L=3.0U W=4.5U
M4 7 6 0 1 CMOSN L=3.0U W=4.5U
M5 9 8 0 1 CMOSN L=3.0U W=4.5U
M6 0 10 8 1 CMOSN L=3.0U W=4.5U
M7 11 4 0 1 CMOSN L=3.0U W=4.5U
M8 10 7 11 1 CMOSN L=3.0U W=4.5U
M9 12 4 7 1 CMOSN L=3.0U W=4.5U
M10 4 7 12 1 CMOSN L=3.0U W=4.5U
M11 0 4 13 1 CMOSN L=3.0U W=4.5U
M12 14 12 0 1 CMOSN L=3.0U W=4.5U
M13 15 14 0 1 CMOSN L=3.0U W=4.5U
M14 3 2 17 16 CMOSP L=3.0U W=4.5U
M15 4 3 17 16 CMOSP L=3.0U W=4.5U
M16 6 5 17 16 CMOSP L=3.0U W=4.5U
M17 7 6 17 16 CMOSP L=3.0U W=4.5U
M18 9 8 17 16 CMOSP L=3.0U W=4.5U
M19 17 10 8 16 CMOSP L=3.0U W=4.5U
M20 10 4 17 16 CMOSP L=3.0U W=4.5U
M21 17 7 10 16 CMOSP L=3.0U W=4.5U
M22 12 13 7 16 CMOSP L=3.0U W=4.5U
M23 13 7 12 16 CMOSP L=3.0U W=4.5U
M24 17 4 13 16 CMOSP L=3.0U W=4.5U
M25 14 12 17 16 CMOSP L=3.0U W=4.5U
M26 15 14 17 16 CMOSP L=3.0U W=4.5U
VIN2 2 0 PULSE (5 0 13NS 2NS 2NS 13NS 30NS)
VIN5 5 0 5
C27 17 0 354.0F
C28 7 0 116.0F
.TRAN 0.2NS 40NS
.PLOT TRAN V(7) V(4) V(14) V(8) (0,5)
.MODEL CMOSN NMOS LEVEL=2.0 LD=0.28U TOX=500.0E-10
+NSUB=1.0E+16 VTO=0.827125 KP=3.286649E-05 GAMMA=1.35960
+PHI=0.6 UO=200.0 UEXP=1.001E-03 UCRIT=999000.0 MJSW=0.33
+DELTA=1.2405 VMAX=100000.0 XJ=0.4U LAMBDA=1.604983E-02
+NSF=1.234795E+12 NEFF=1.001E-02 NSS=0.0E+00 TPG=1.0 RSH=25
+CGSO=5.2E-10 CGDO=5.2E-10 CJ=3.2E-4 MJ=0.5 CJSW=9E-10
.MODEL CMOSP PMOS LEVEL=2.0 LD=0.28U TOX=500.0E-10
+NSUB=1.121088E+14 VTO=-0.894654 KP=1.526452E-05 MJSW=0.33
+PHI=0.6 UO=100.0 UEXP=0.153411 UCRIT=16376.5 GAMMA=0.879003
+DELTA=1.93831 VMAX=100000.0 XJ=0.4U LAMBDA=4.708659E-02
+NSF=8.788617E+11 NEFF=1.001E-02 NSS=0.0E+00 TPG=-1.0 RSH=95
+CGSO=4.0E-10 CGDO=4.0E-10 CJ=2.0E-4 MJ=0.5 CJSW=4.5E-10
.END
```

Figure A.3 Sample SPICE Deck.

APPENDIX B
ORGANELLE DESIGNS

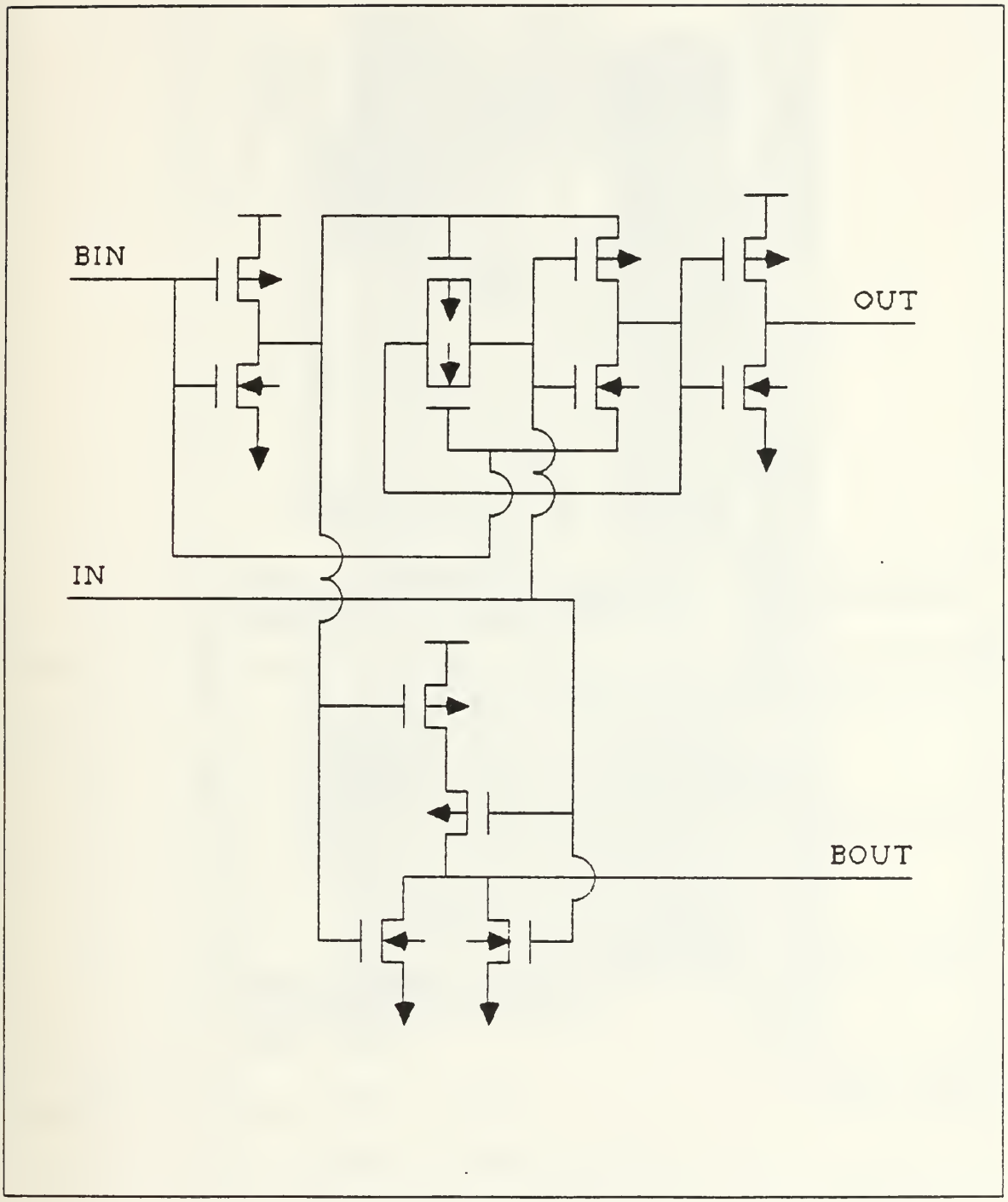

Figure B.1 Decrementor Transistor Schematic.

. TABLE 8
DECREMENTOR TRUTH TABLE

IN	BIN	OUT	BOUT
0	0	0	0
1	0	1	0
1	1	0	0
0	1	1	1

TABLE 9
DECREMENTOR TIMING DATA

PARAMETER	FANOUT	t_{dr}	t_{df}	t_r	t_f
IN → OUT	1	3.0ns	2.4ns	1.7ns	1.3ns
	4	3.5ns	2.8ns	2.5ns	2.0ns
IN → BOUT	1	1.4ns	1.3ns	2.7ns	2.8ns
	4	2.4ns	2.2ns	4.3ns	3.8ns
BIN → OUT	1	2.2ns	2.1ns	1.4ns	1.6ns
	4	2.6ns	2.5ns	2.3ns	2.4ns
BIN → BOUT	1	2.3ns	2.1ns	2.0ns	1.6ns
	4	3.2ns	2.6ns	3.9ns	2.5ns

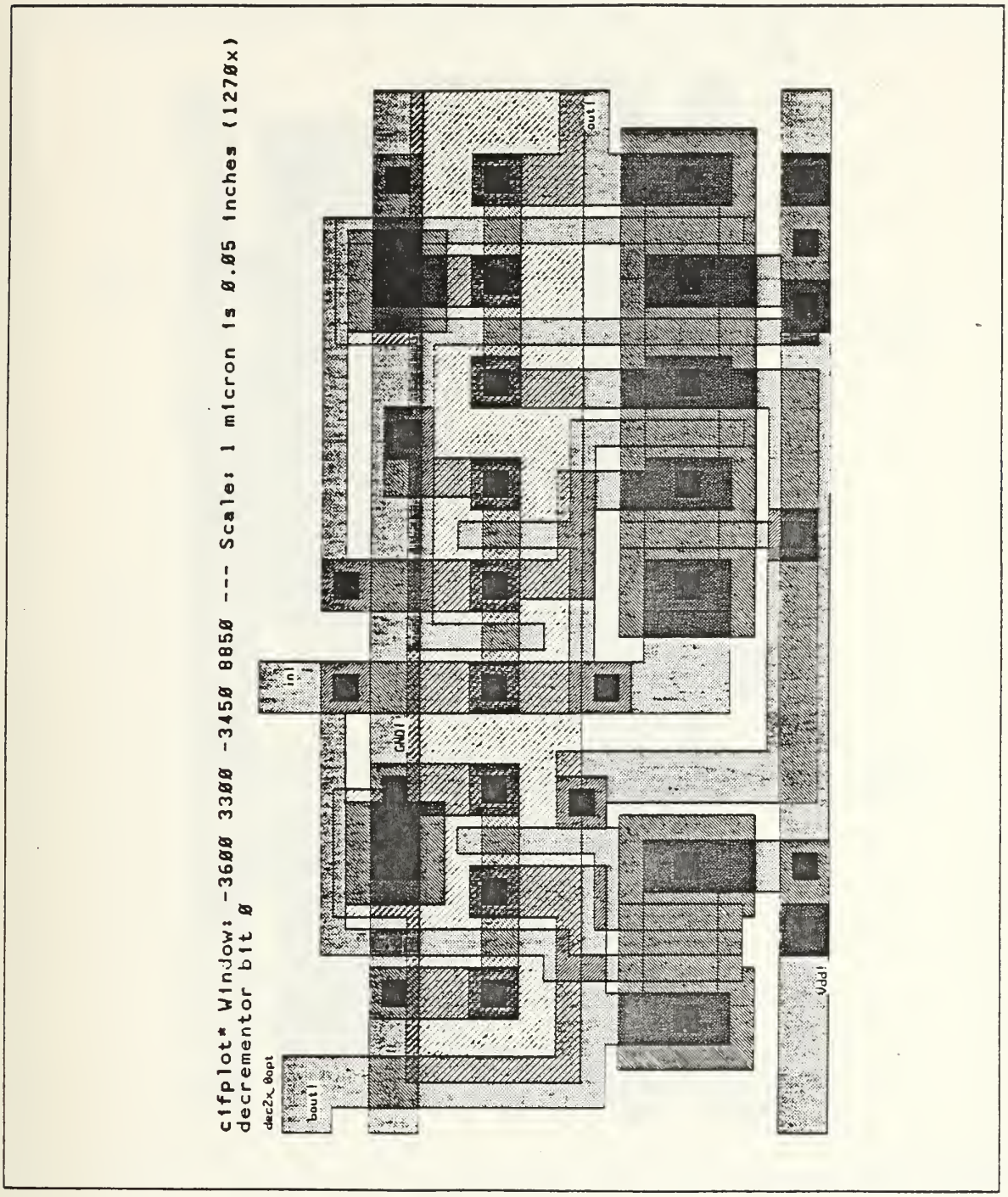

Figure B.2 Decrementor Bit 0.

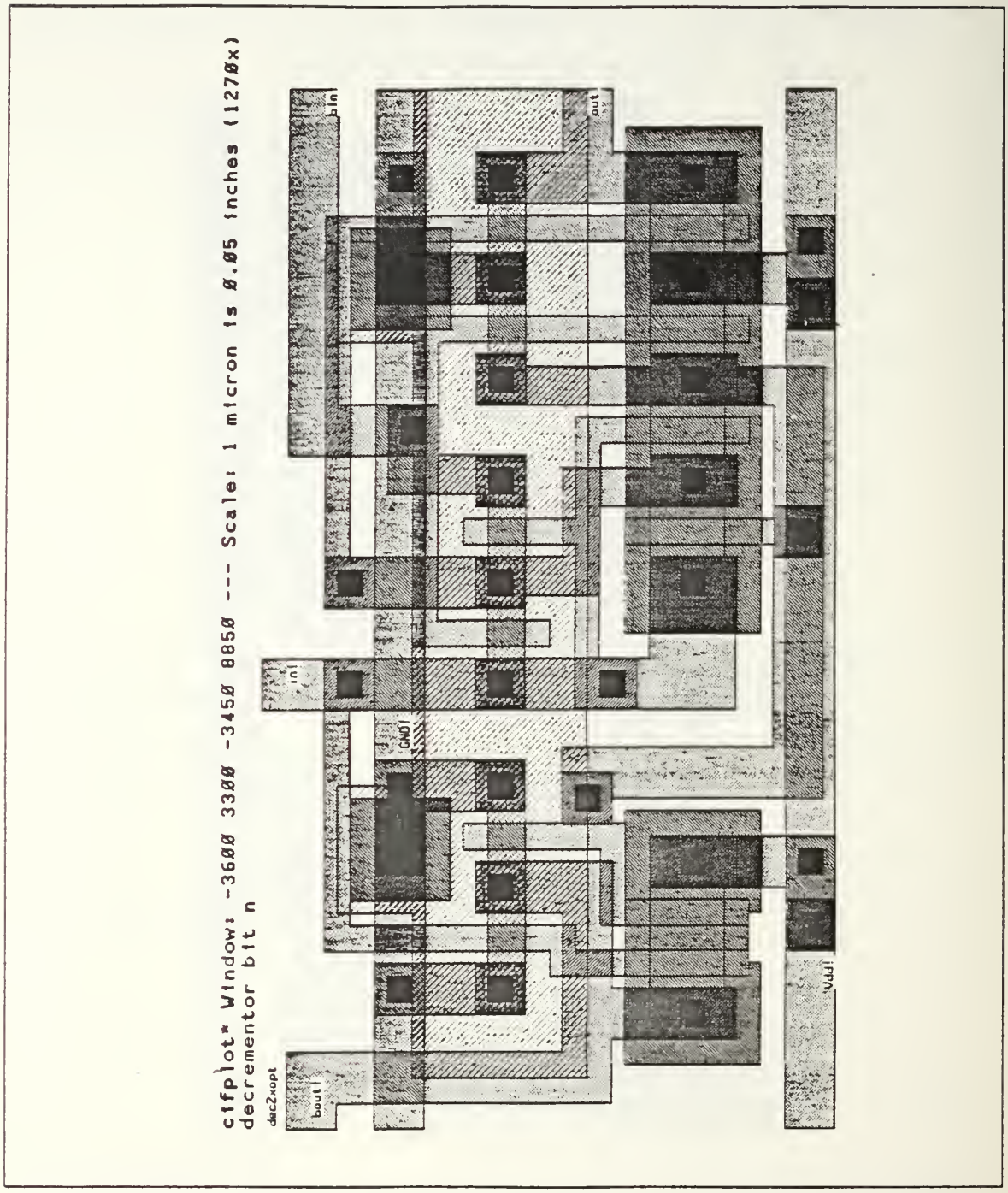

Figure B.3 Decrementor Bit n.

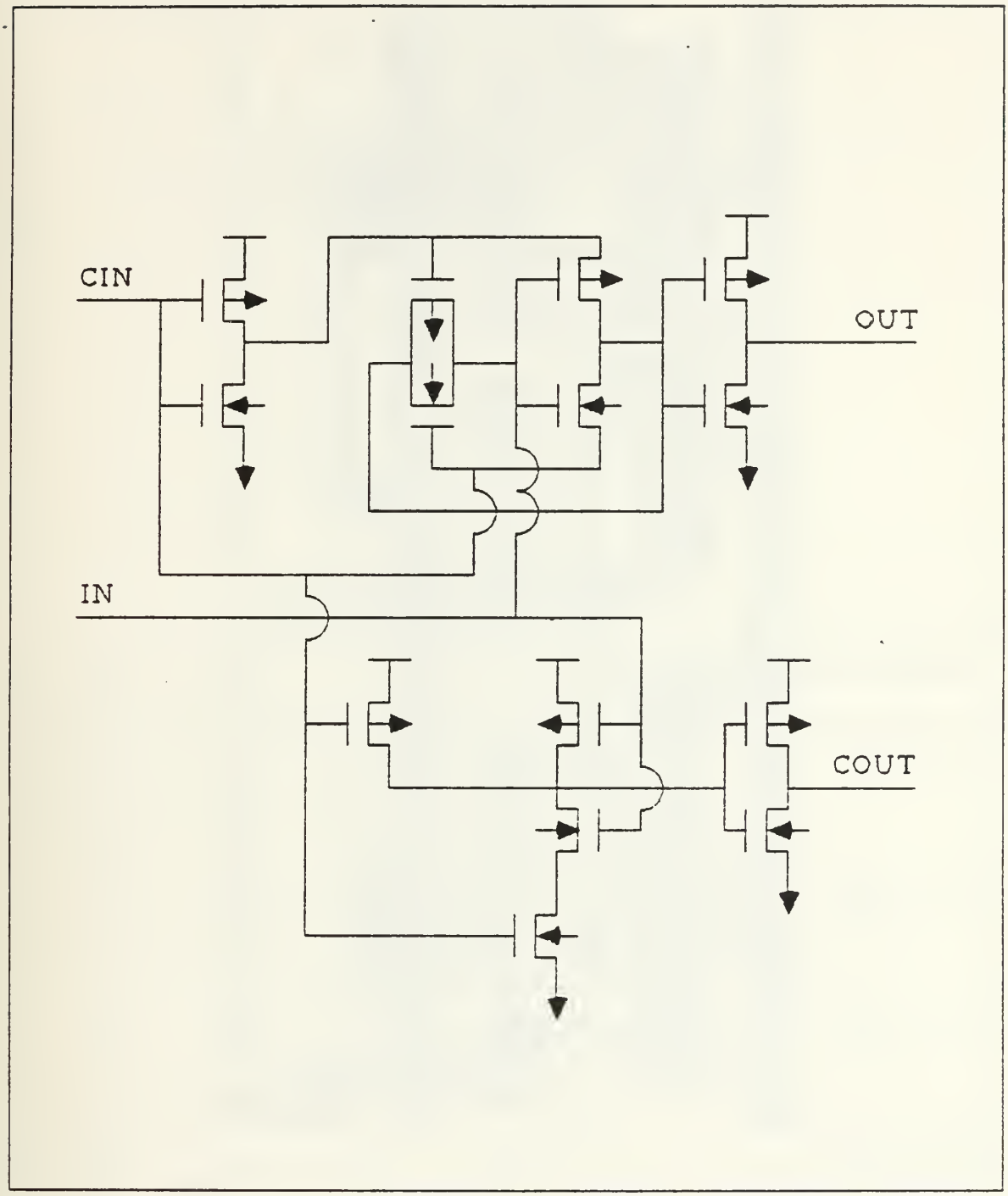

Figure B.4 Incrementor Transistor Schematic.

TABLE 10
INCREMENTOR TRUTH TABLE

IN	CIN	OUT	COUT
0	0	0	0
1	0	1	0
1	1	0	1
0	1	1	0

TABLE 11
INCREMENTOR TIMING DATA

PARAMETER	FANOUT	t_{dr}	t_{df}	t_r	t_f
IN → OUT	1	2.4ns	2.7ns	2.0ns	1.4ns
	4	3.2ns	3.3ns	3.8ns	2.1ns
IN → COUT	1	2.4ns	2.0ns	2.1ns	1.2ns
	4	3.3ns	2.4ns	3.7ns	1.8ns
CIN → OUT	1	1.7ns	2.7ns	1.6ns	1.3ns
	4	2.3ns	3.4ns	3.6ns	2.0ns
CIN → COUT	1	1.9ns	1.9ns	1.7ns	1.0ns
	4	2.7ns	2.3ns	3.4ns	1.7ns

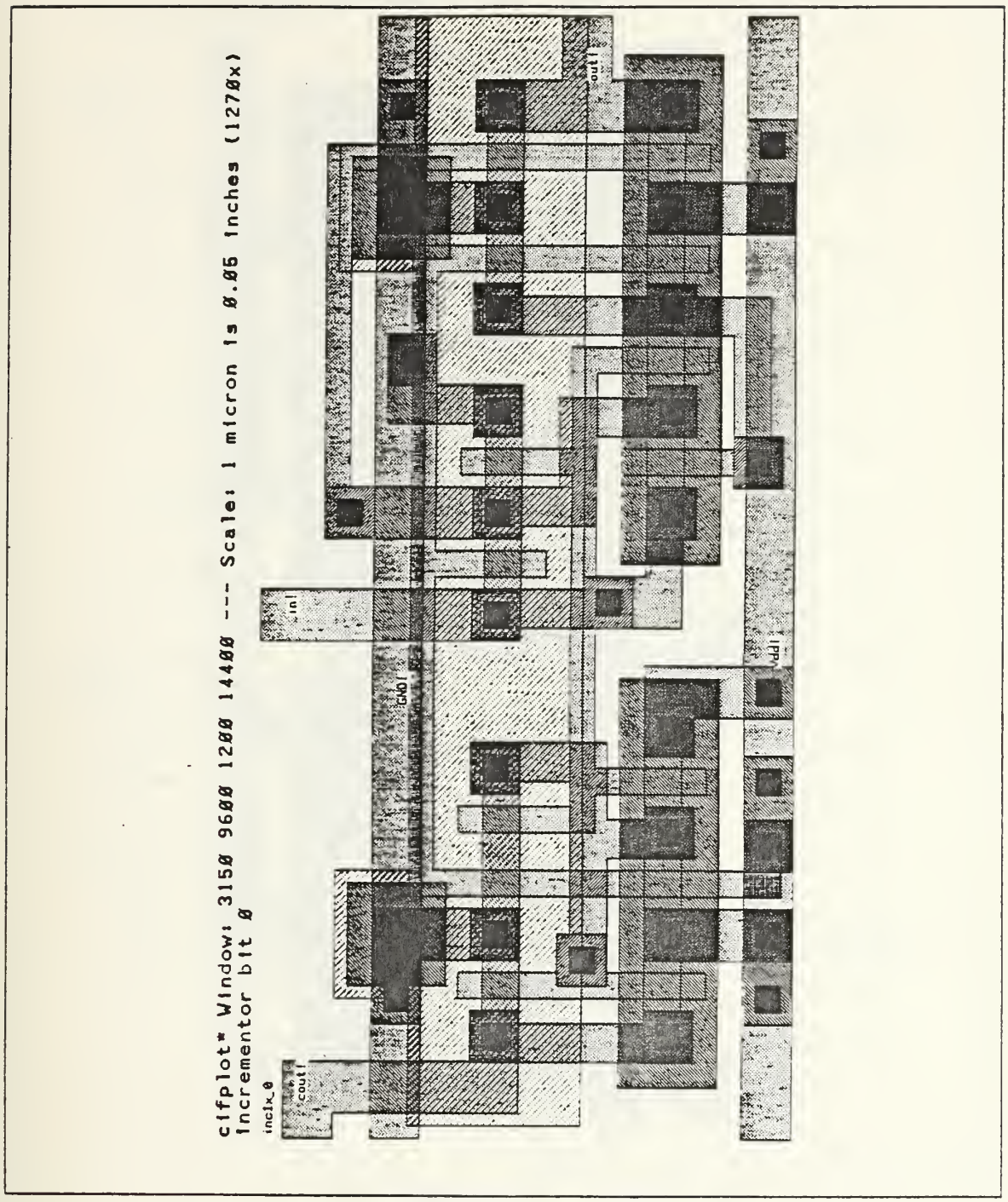

Figure B.5 Incrementor Bit 0.

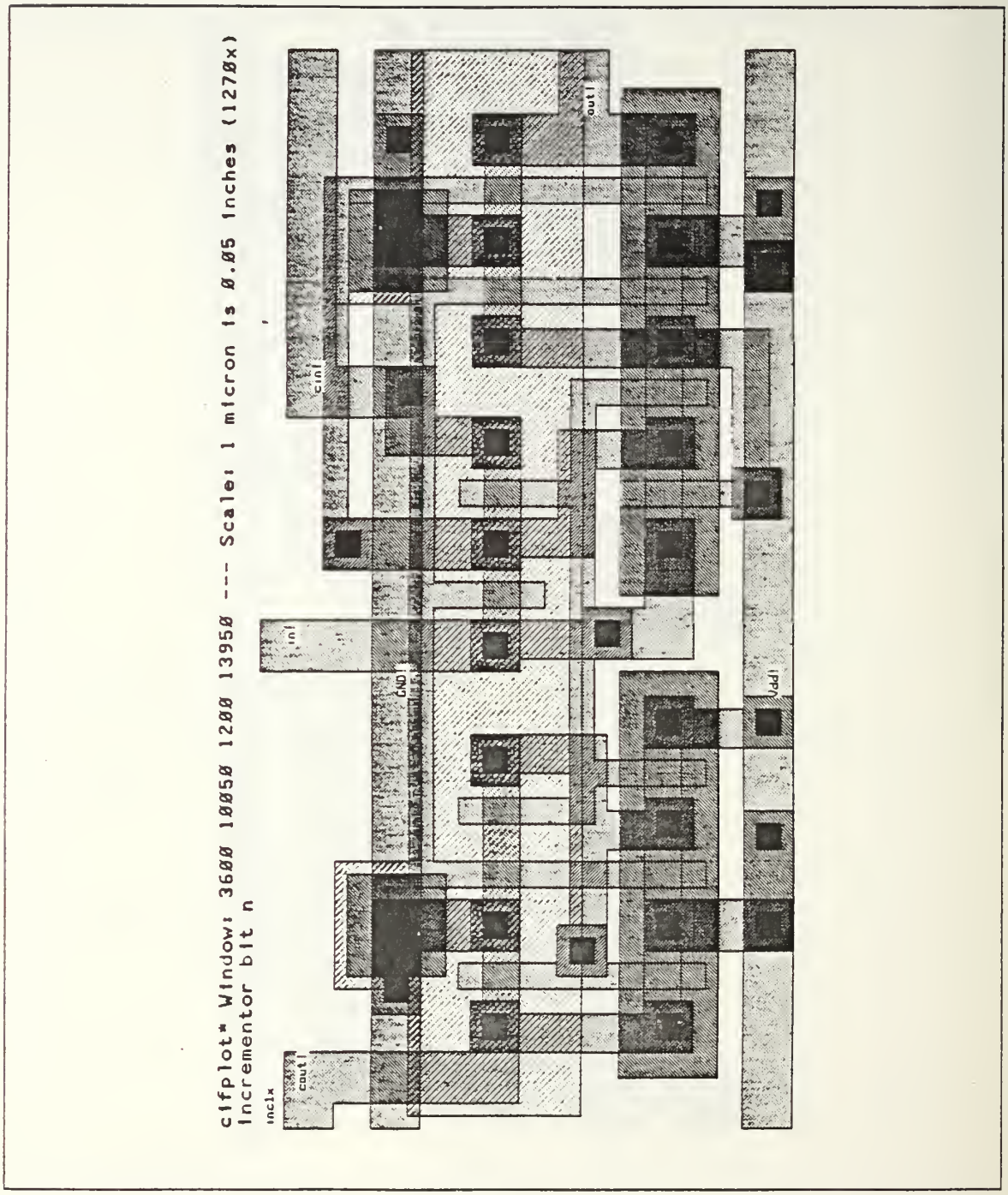

Figure B.6 Incrementor Bit n.

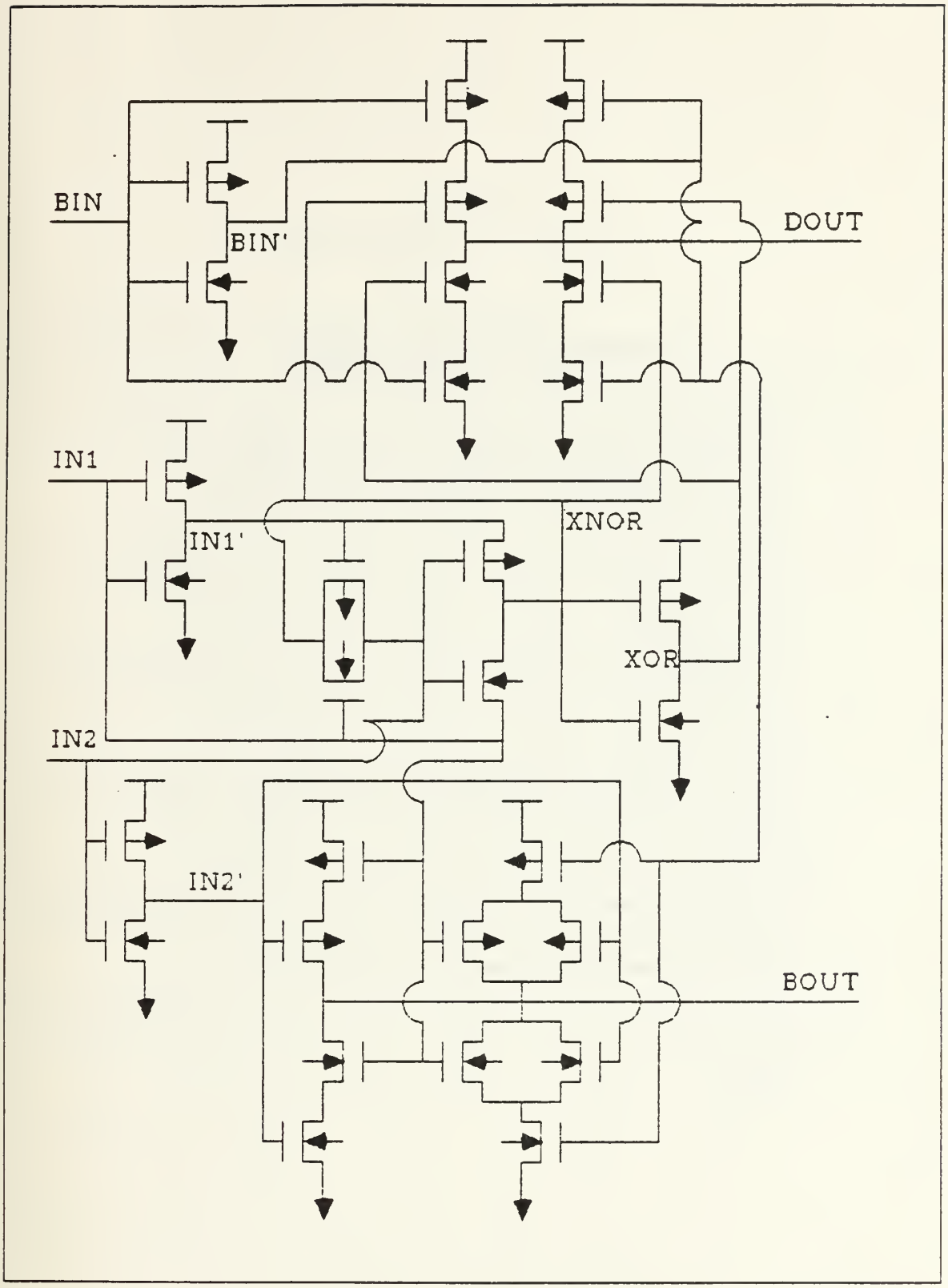

Figure B.7 Subtractor Transistor Schematic.

TABLE 12
SUBTRACTOR TRUTH TABLE

IN1	IN2	BIN	DOUT	BOUT
0	0	0	0	0
0	0	1	1	1
0	1	0	1	1
0	1	1	0	1
1	0	0	1	0
1	0	1	0	0
1	1	0	0	0
1	1	1	1	1

TABLE 13
SUBTRACTOR TIMING DATA

PARAMETER	FANOUT	t_{dr}	t_{df}	t_r	t_f
IN1 → DOUT	1	3.4ns	2.8ns	1.6ns	1.6ns
	4	4.0ns	3.4ns	2.9ns	2.7ns
IN1 → BOUT	1	3.8ns	3.4ns	5.9ns	6.3ns
	4	4.6ns	4.2ns	7.6ns	7.7ns
IN2 → DOUT	1	3.0ns	2.8ns	.1.8ns	1.7ns
	4	3.8ns	3.5ns	3.4ns	3.1ns
IN2 → BOUT	1	4.8ns	4.6ns	6.4ns	5.6ns
	4	5.5ns	5.4ns	8.0ns	7.2ns
BIN → DOUT	1	2.0ns	1.8ns	1.6ns	1.7ns
	4	2.6ns	2.4ns	3.1ns	2.9ns
BIN → BOUT	1	4.1ns	3.7ns	5.6ns	5.6ns
	4	4.8ns	4.4ns	7.3ns	7.2ns

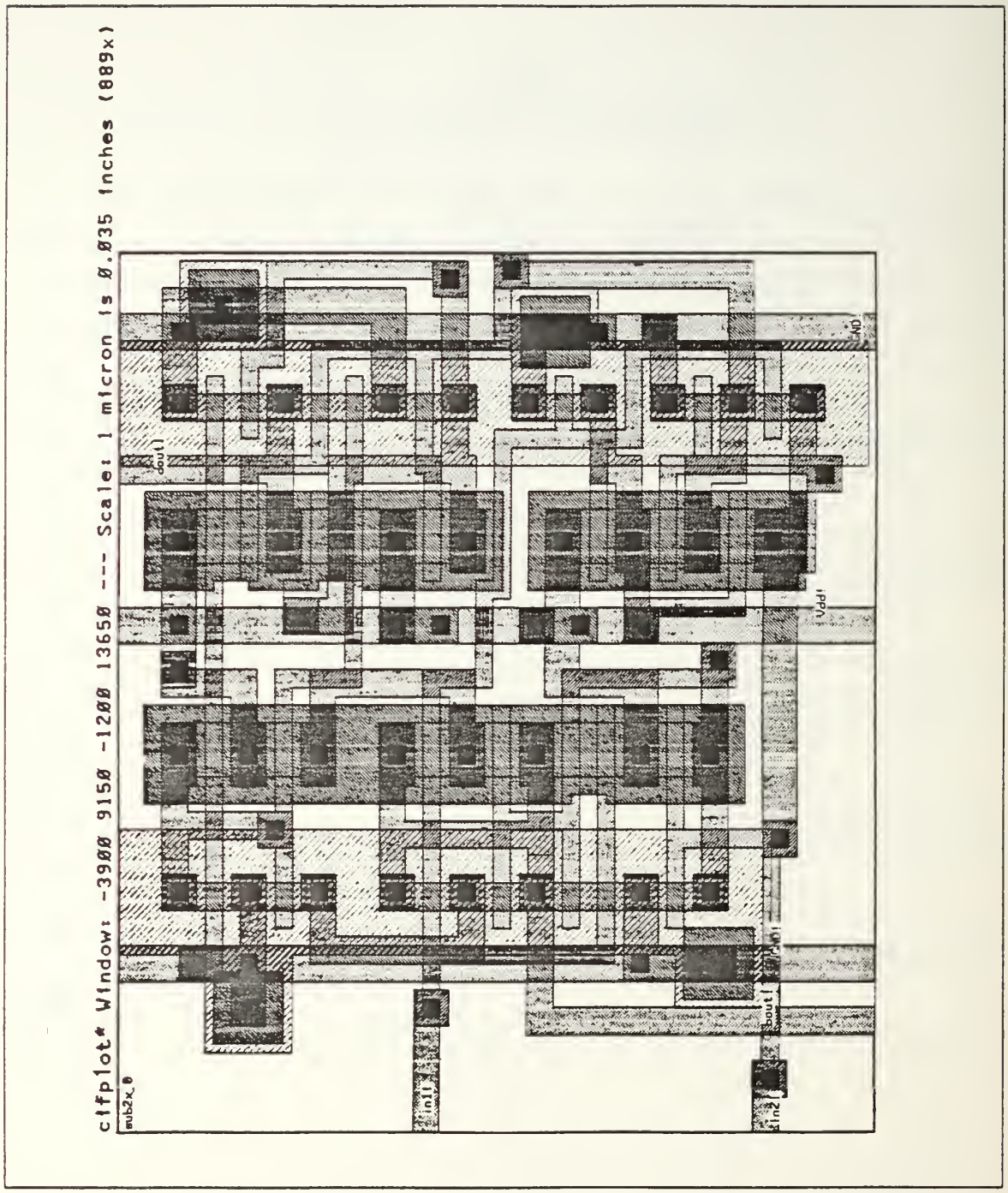

Figure B.8 Subtractor Bit 0.

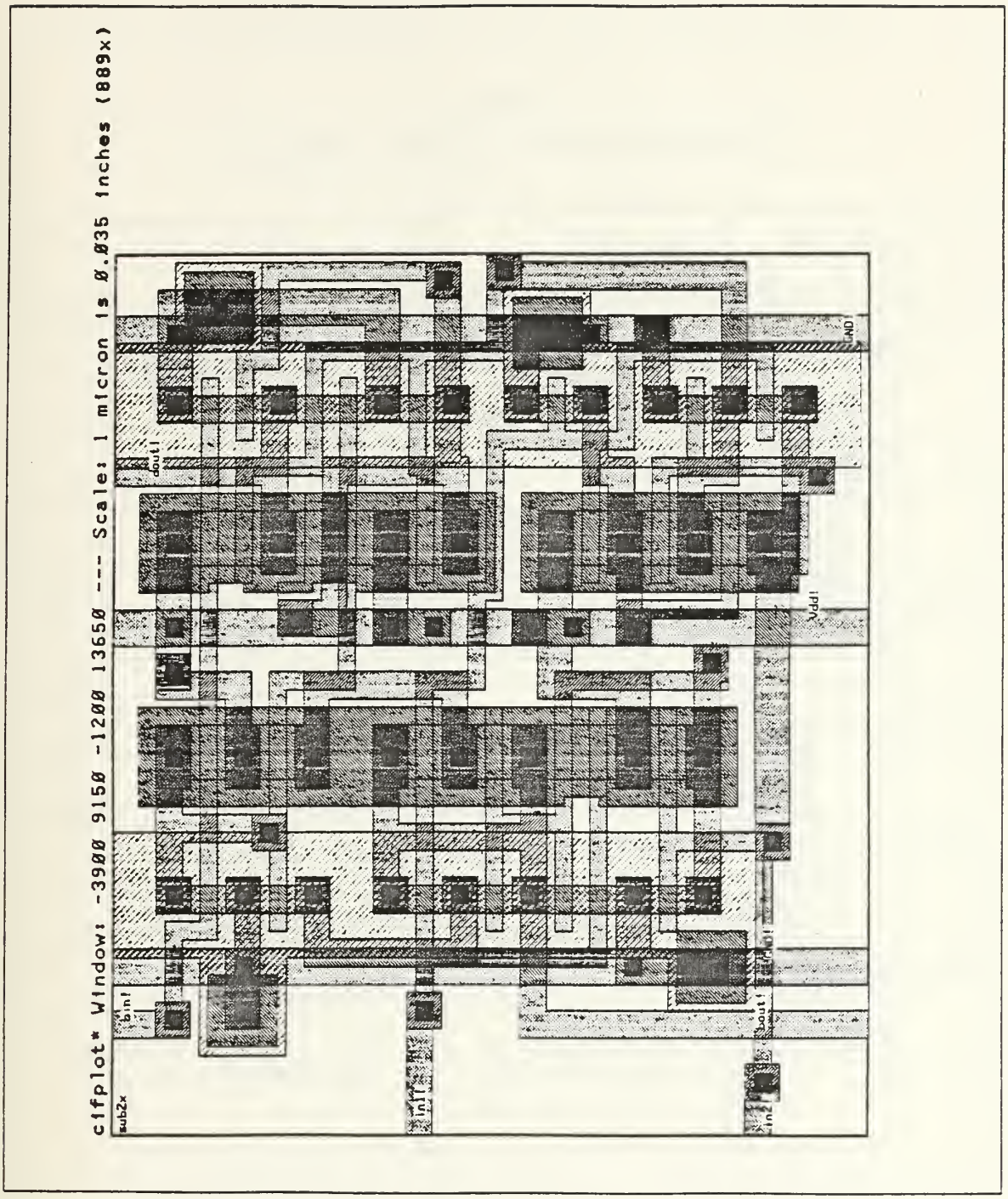

Figure B.9 Subtractor Bit n.

TABLE 14
SUBTRACTOR TIMING DATA - VERSION 1

PARAMETER	FANOUT	t_{dr}	t_{df}	t_r	t_f
IN1 → DOUT	1	3.4ns	2.8ns	1.6ns	1.6ns
	4	4.0ns	3.4ns	3.0ns	2.8ns
IN1 → BOUT	1	4.2ns	4.1ns	7.2ns	7.3ns
	4	5.0ns	4.8ns	9.2ns	9.0ns
IN2 → DOUT	1	3.0ns	2.8ns	1.8ns	1.8ns
	4	3.8ns	3.4ns	3.3ns	3.1ns
IN2 → BOUT	1	5.4ns	4.5ns	7.8ns	6.7ns
	4	6.2ns	5.4ns	9.5ns	8.4ns
BIN → DOUT	1	2.0ns	1.8ns	1.6ns	1.7ns
	4	2.6ns	2.4ns	3.0ns	2.9ns
BIN → BOUT	1	4.6ns	4.3ns	6.9ns	6.9ns
	4	5.4ns	4.9ns	8.6ns	8.5ns

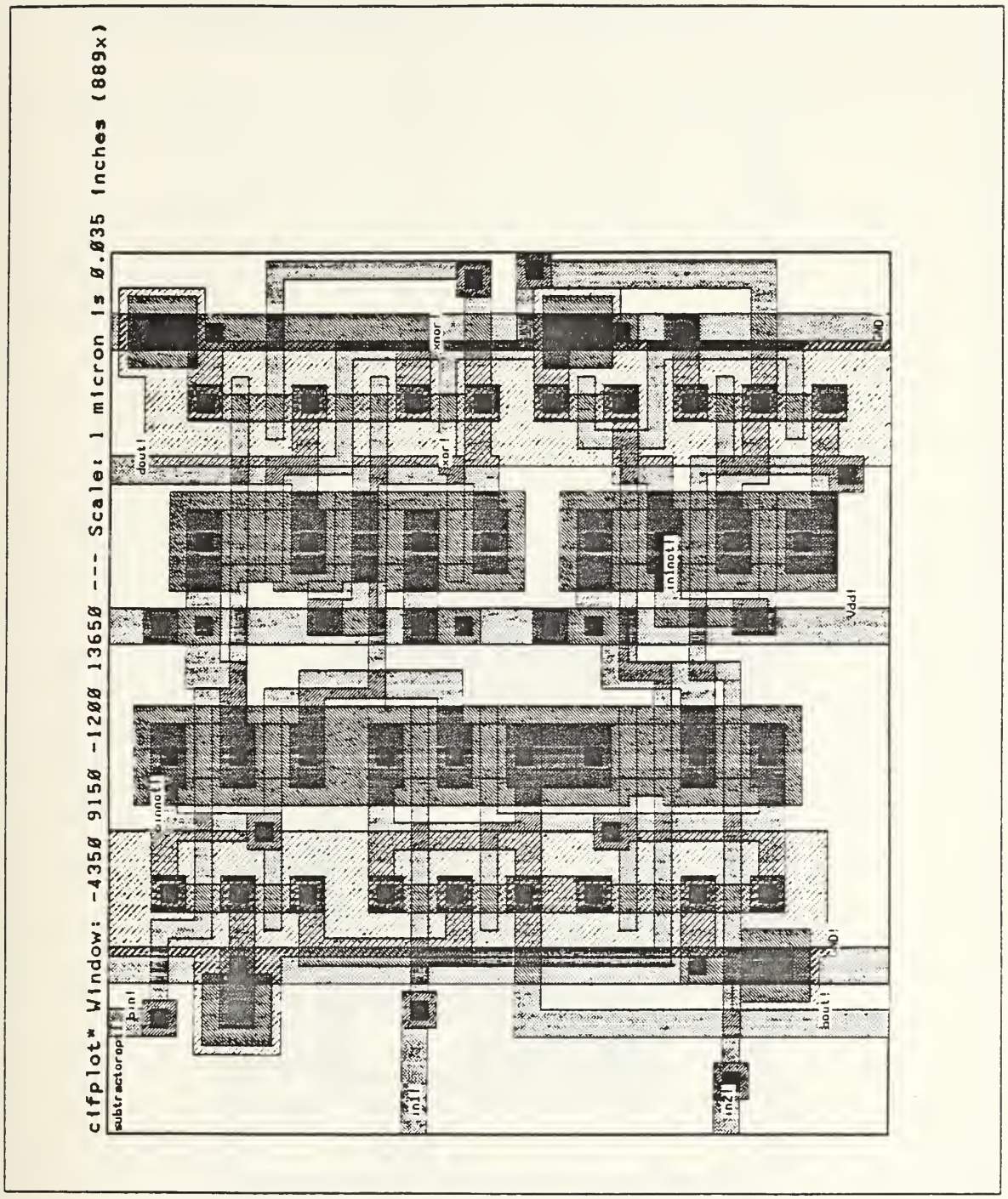

Figure B.10 Subtractor Bit n - Version 1 (not used).

APPENDIX C
MACPITTS CHIP LAYOUTS

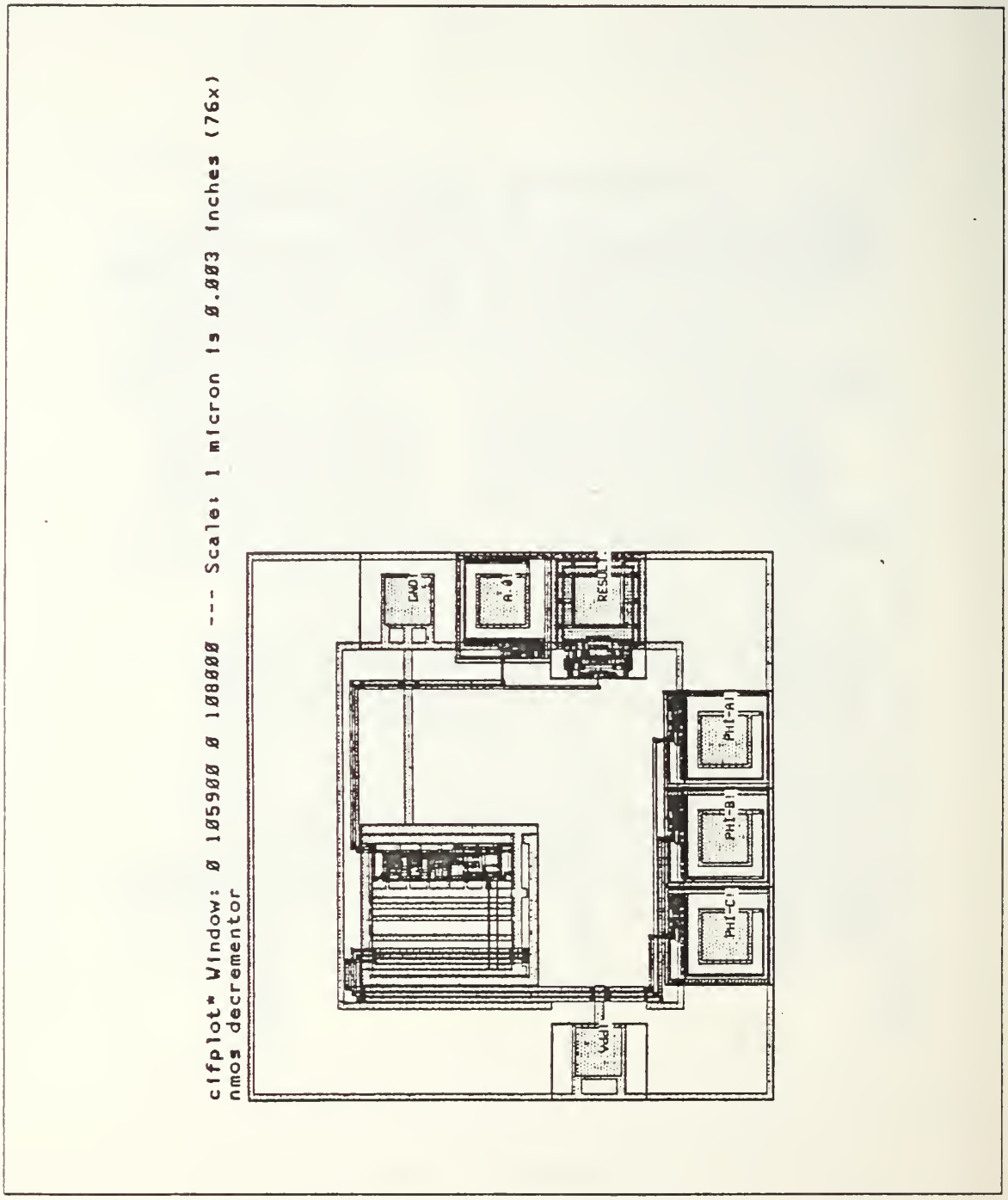

Figure C.1 NMOS Decrementor Chip (One Bit Word).

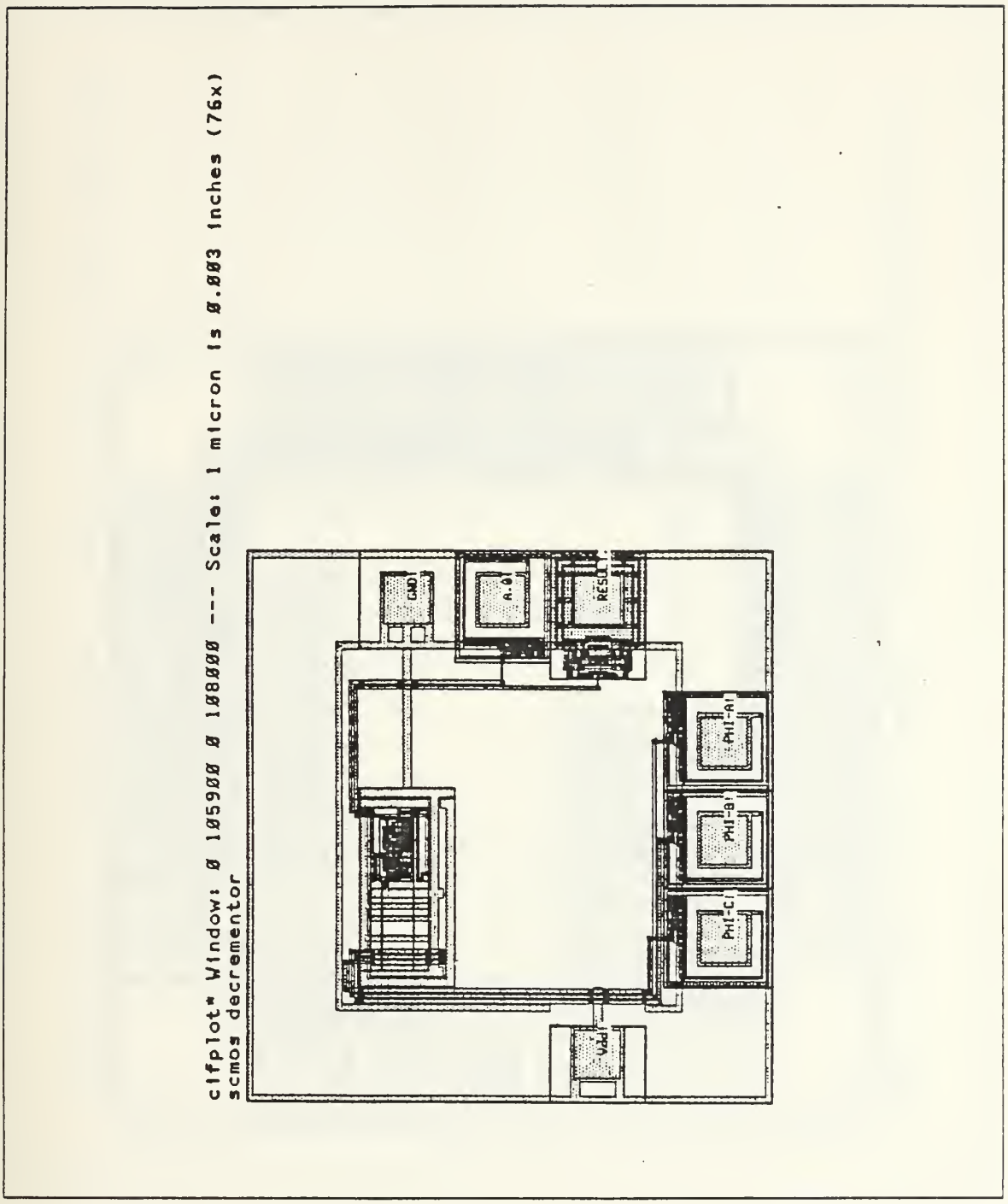

Figure C.2 SCMOS Decrementor Chip (One Bit Word).

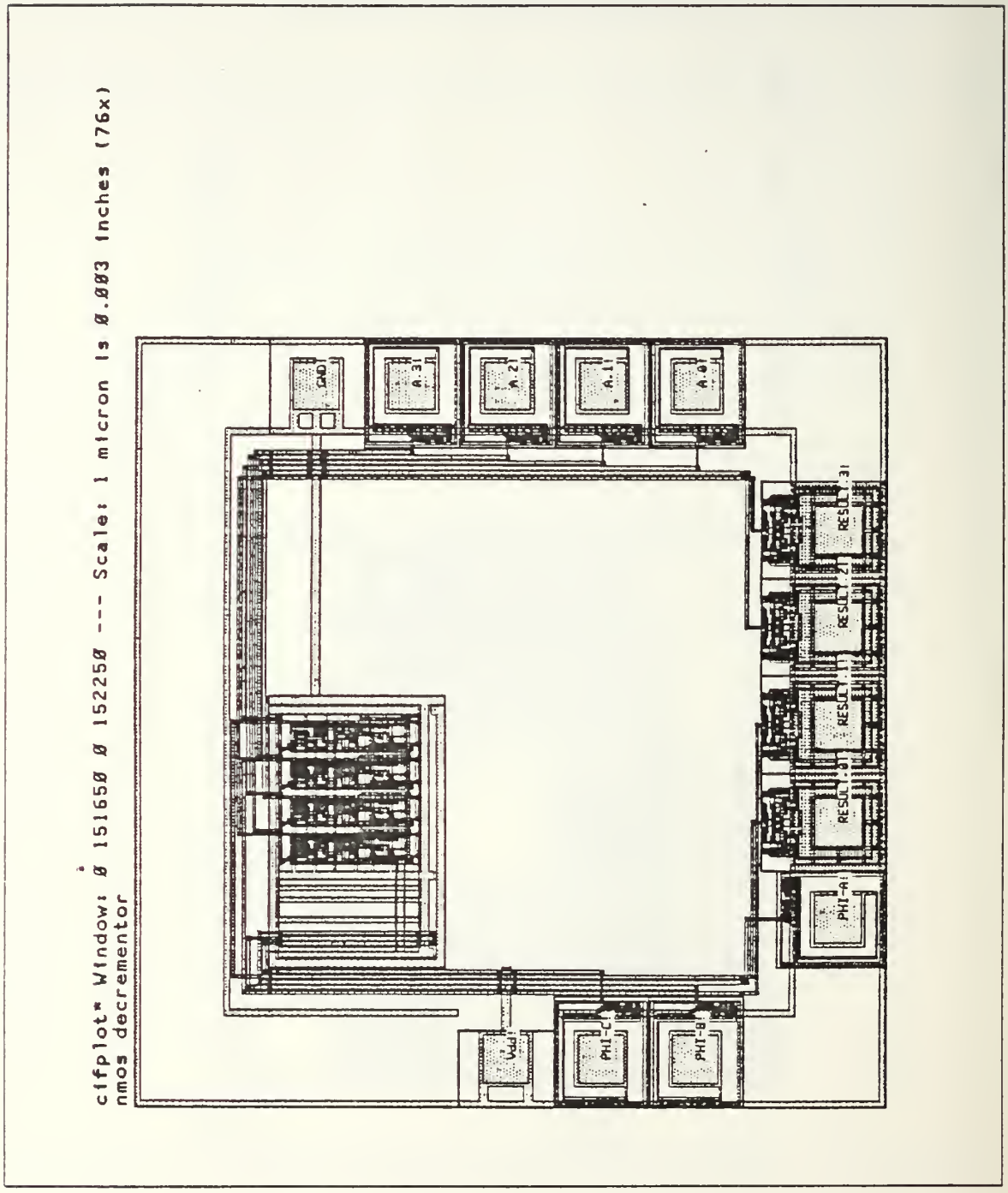

Figure C.3 NMOS Decrementor Chip (Four Bit Word).

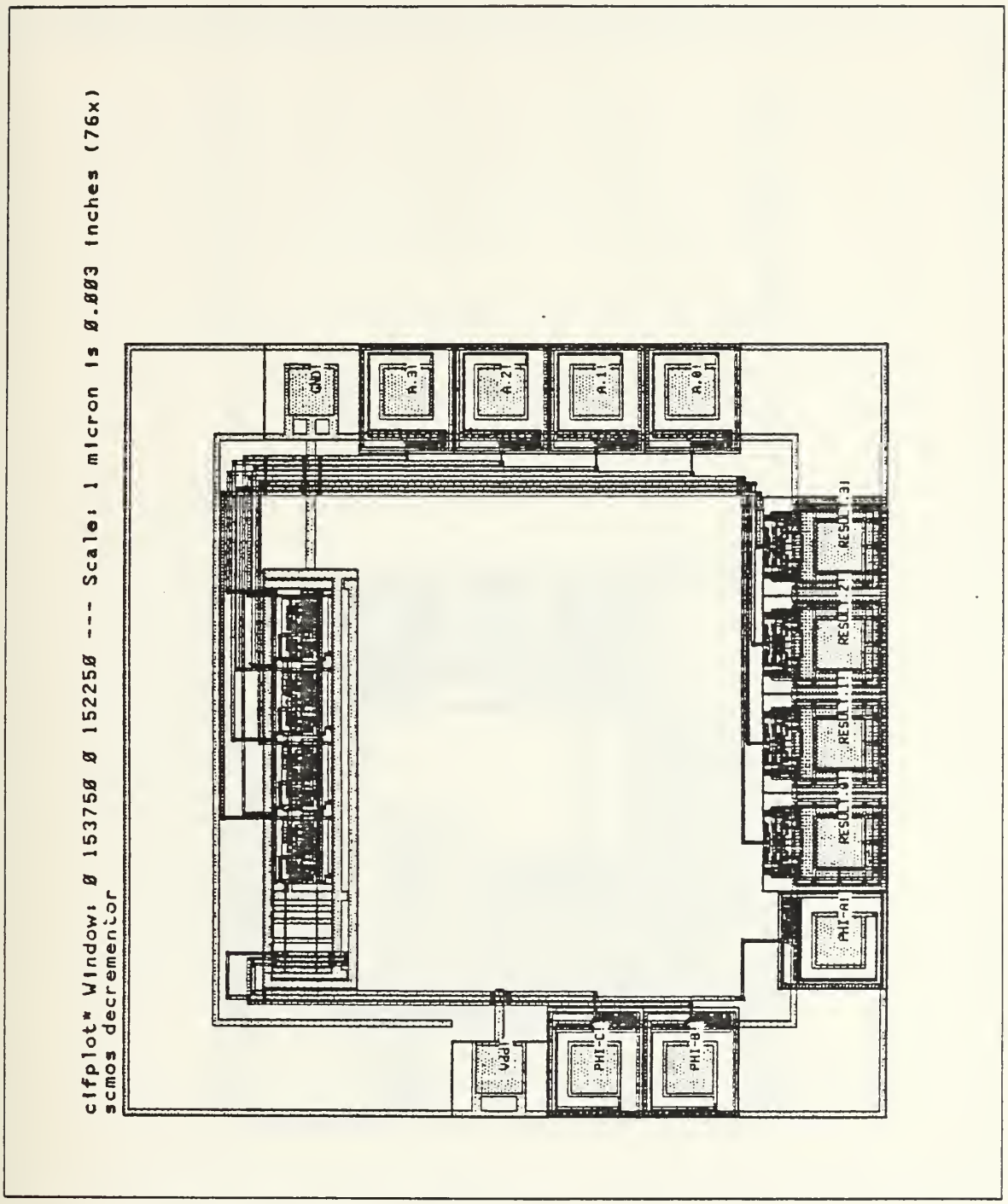

Figure C.4 SCMOS Decrementor Chip (Four Bit Word).

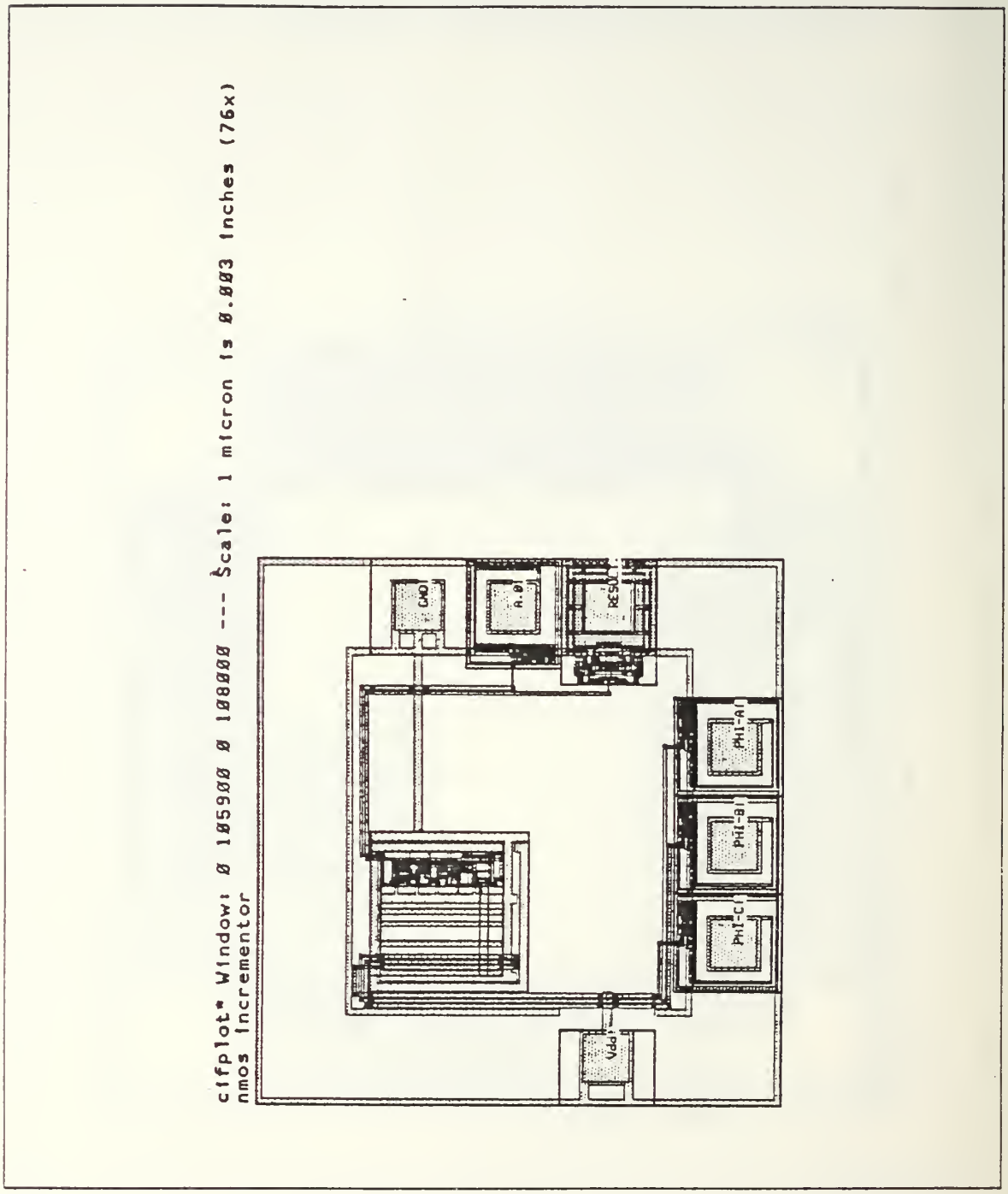

Figure C.5 NMOS Incrementor Chip (One Bit Word).

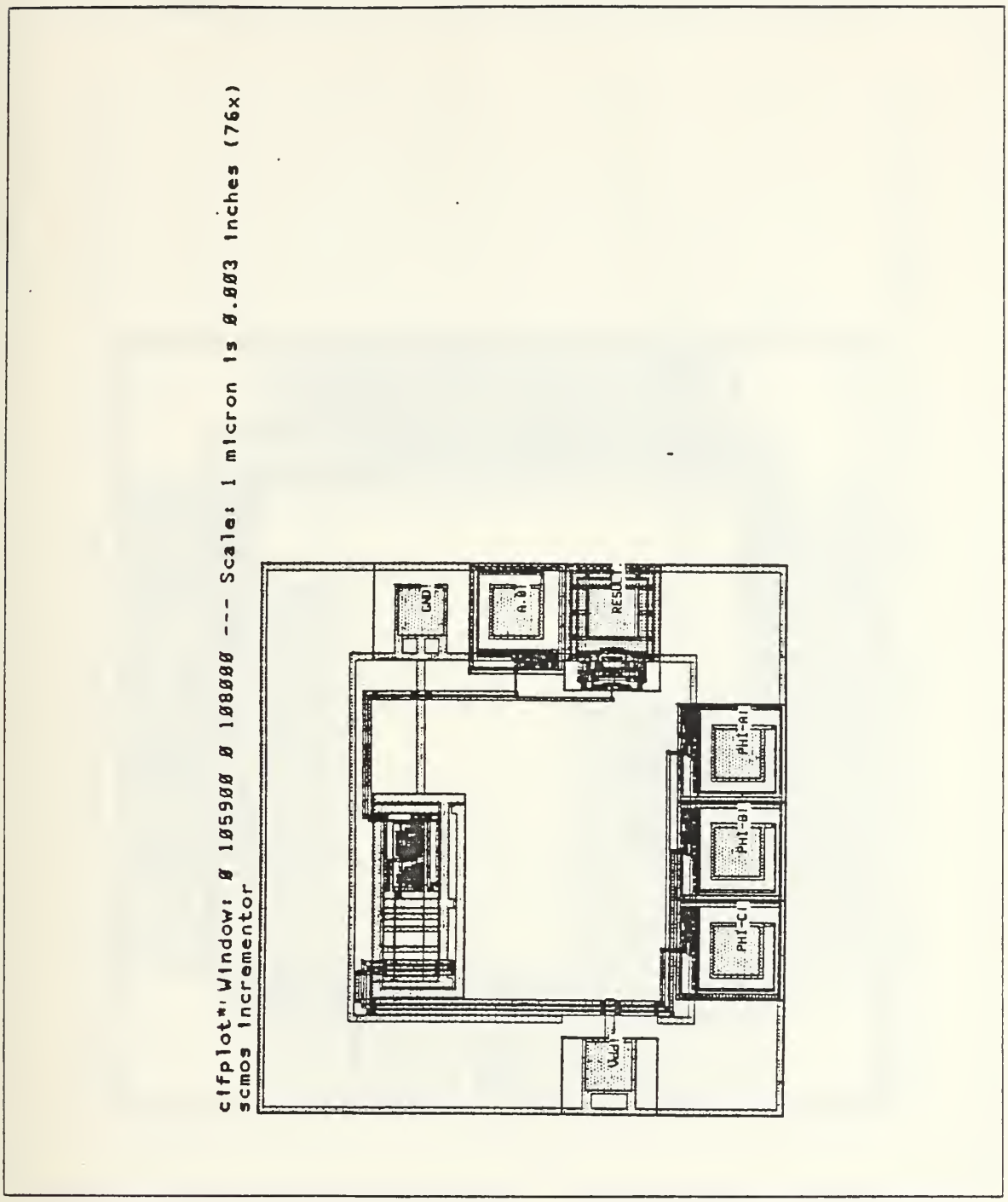

Figure C.6 SCMOS Incrementor Chip (One Bit Word).

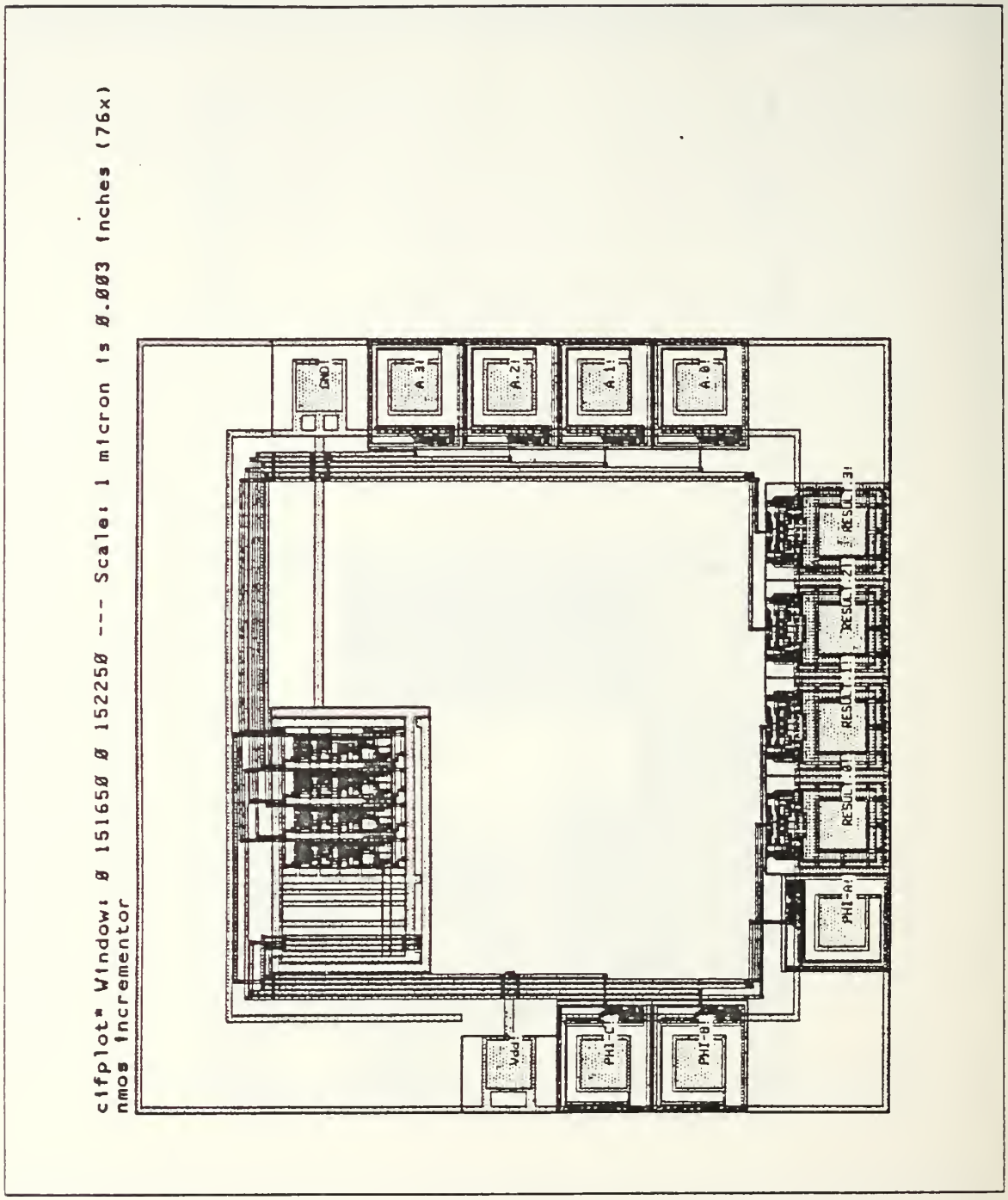

Figure C.7 NMOS Incrementor Chip (Four Bit Word).

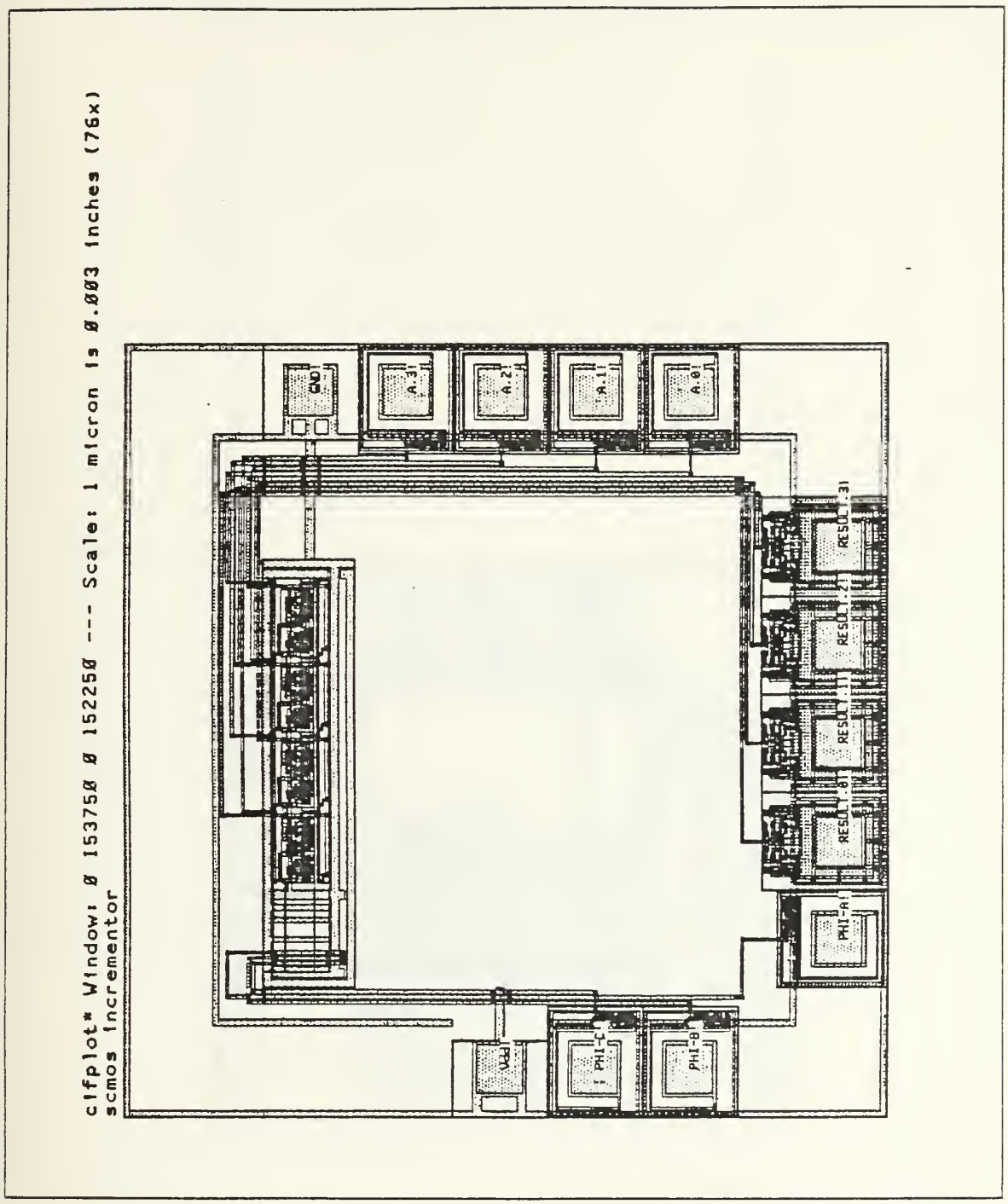

Figure C.8 SCMOS Incrementor Chip (Four Bit Word).

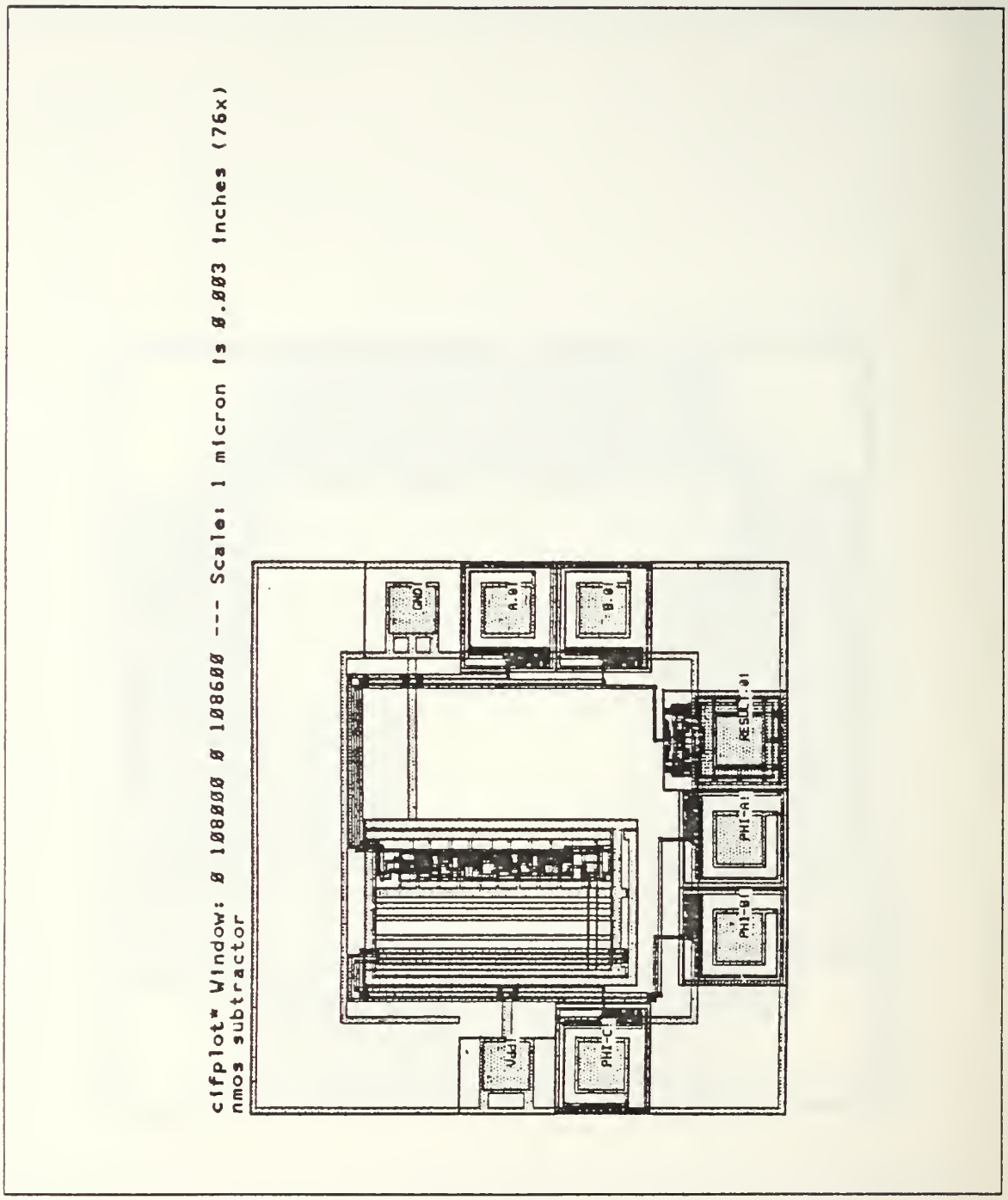

Figure C.9 NMOS Subtractor Chip (One Bit Word).

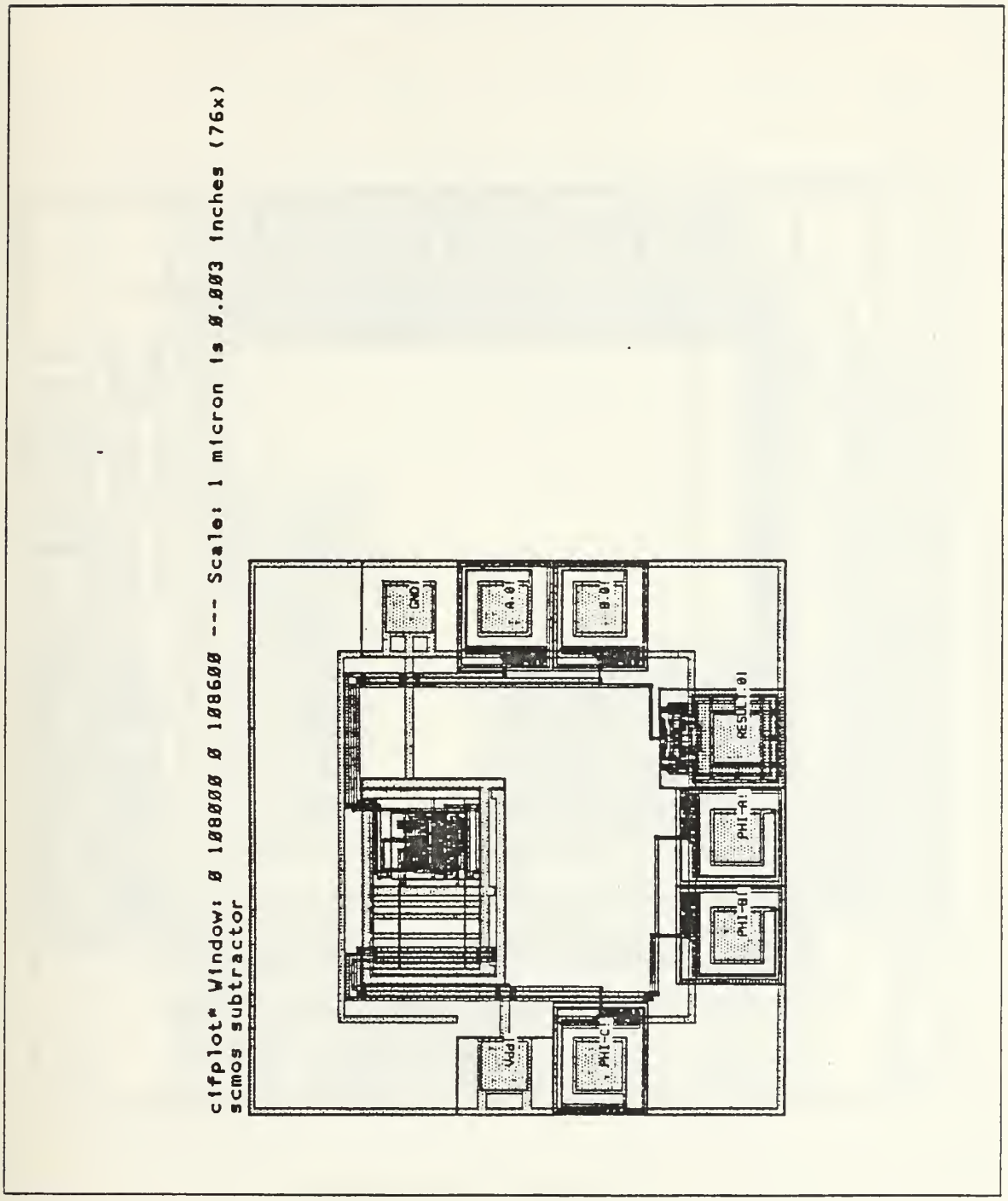

Figure C.10 SCMOS Subtractor Chip (One Bit Word).

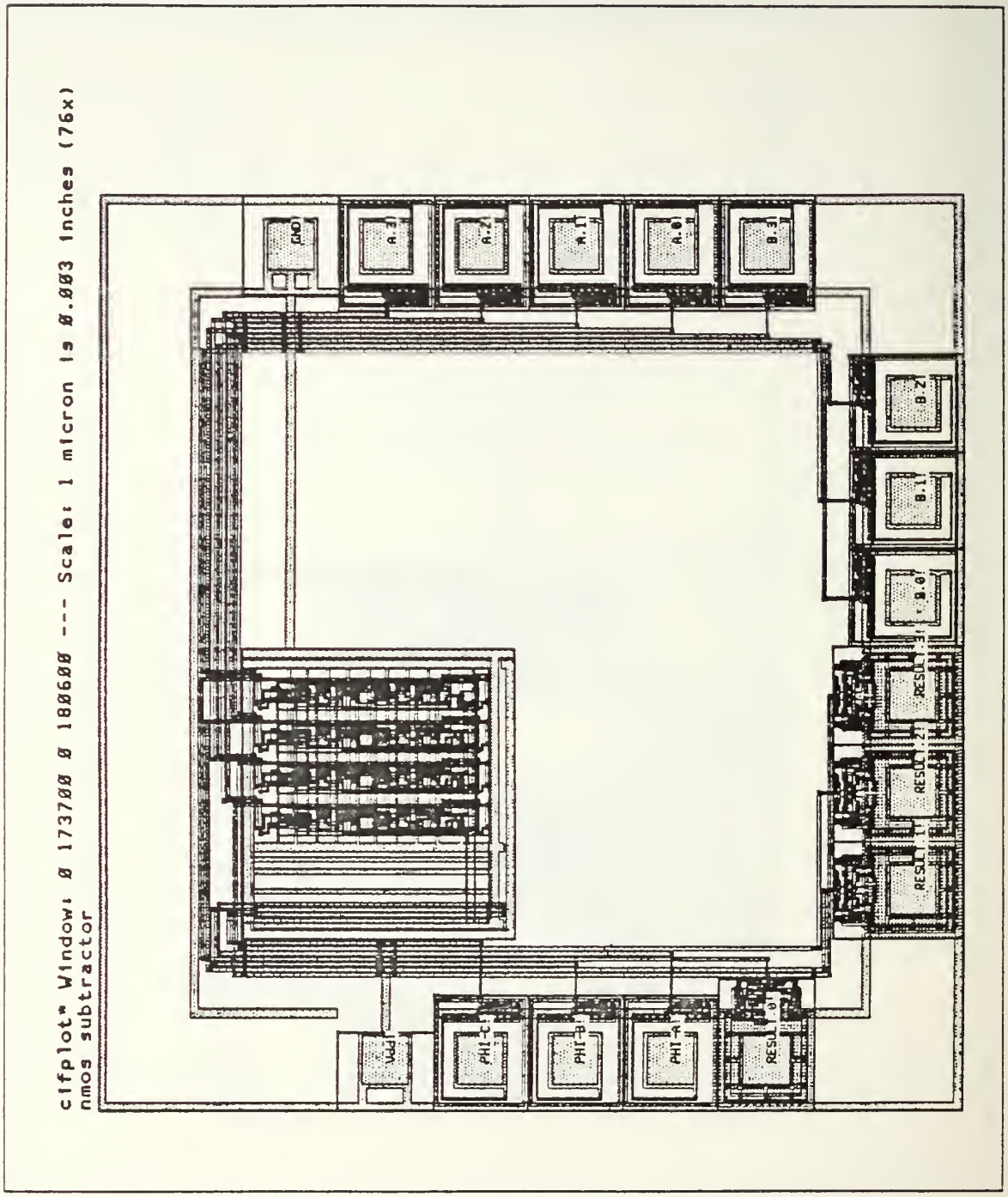

Figure C.11 NMOS Subtractor Chip (Four Bit Word).

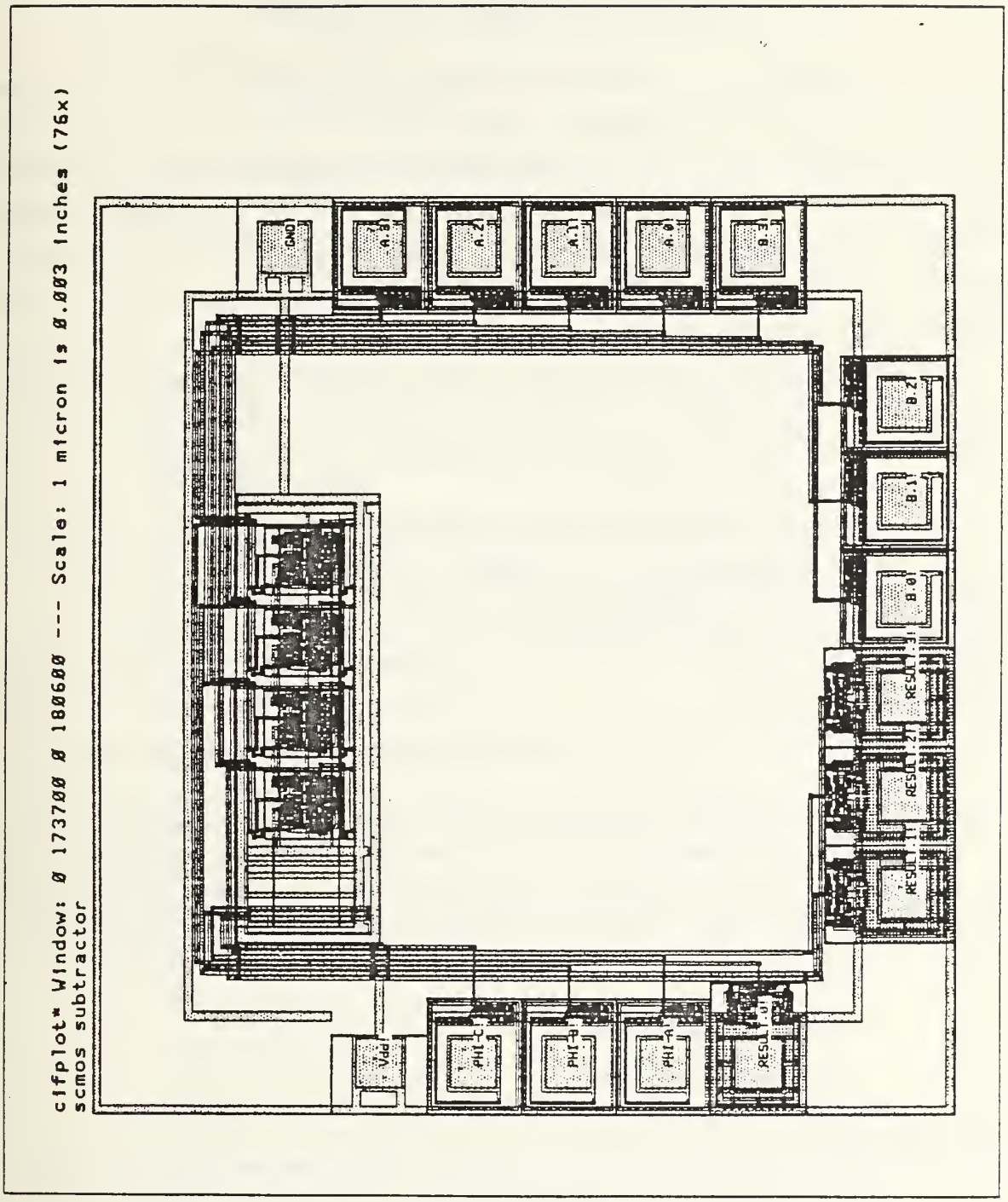

Figure C.12 SCMOS Subtractor Chip (Four Bit Word).

APPENDIX D
SCMOS ORGANELLE INSERTION

The insertion of an SCMOS organelle into the MacPitts Silicon Compiler installed on the CS-VAX computer is described by E. Malagon [Ref. 7: p. 62]. The procedure outlined below describes the insertion of an organelle into the MacPitts Silicon Compiler installed on the ISI workstations. The ISI environment contains changes made to the compiler by J. Harmon [Ref. 8], and the SCMOS organelles inserted into the CS-VAX MacPitts environment by E. Malagon [Ref. 7]. The following procedures are described:

- conversion of the CIF code of the organelle designed in the Magic environment to L5 code
- insertion of the L5 code into the organelles.l file
- generation of the organelles.o file
- creation and insertion of the library file entry for the organelle
- creation of the MacPitts input program, the .mac file
- generation of the MacPitts chip layout

In the following procedure <filename> should be replaced by the name of the organelle to be inserted.

1. Convert the CIF code of the organelle designed in the Magic environment to L5 code.
    a. Enter the Magic environment on an ISI workstation, and create the .cif file. Make a final check at this time of the connection point extensions, to be sure that they are the required distance beyond the substrate contacts, vias, transistors, etc, as indicated in the design guidelines. To create the .cif file enter:
        - %magic
        - :load <filename>
        - :cif write <filename>
        - :q
    b. If the created <filename>.cif file is not in the 'macpitts' directory, copy it to the 'macpitts' directory. Execute the following command before changing to the 'macpitts' directory:
        - %cp <filename>.cif /path to 'macpitts' directory/<filename>.cif

c. In the 'macpitts' directory, use cifdef.l to convert the CIF code to L5 code:

- %Macpitts (enter the MacPitts environment)
  usage: macpitts <filename> [<option>]
  [Return to top level]
- ->(include cifdef.l)
  [load cifdef.l]
  t
- ->(setq minimum-feature-size '300)
  300
- ->(cifsave '<filename>)
  Reading in symbol 1
  OK t
- ->(exit)

The new file will be <filename>.L5 located in the 'macpitts' directory. The SCMOS L5 code will be a list of 'rect' statements enclosed by the L5 function 'defsymbol'. Figure D.1 shows a portion of the L5 code for the incrementor organelle.

2. Enter the L5 code of the SCMOS organelle into the organelles.l file:

a. Make a working copy of organelles.l for insertion of the new L5 code. A good method is to copy the existing organelles.l file into a file called organelles.old.l. Then move the organelles.l file into a temporary file (e.g. orgtemp.l), as shown below:

- %cp organelles.l organelles.old.l
- %mv organelles.l orgtemp.l

You now will have two copies of the existing organelle.l file, one copy in organelles.old.l which will not be changed, and one temporary copy. The following instruction assume these two copies have been made.

b. Append the new L5 code (<filename>.L5) to the end of the temporary copy and put this into the new organelles.l file using the following command:

- %cat orgtemp.l <filename>.L5 > organelles.l
- %rm orgtemp.l
- rm: remove orgtemp.l? y

This method of concatenating the new SCMOS code to the temporary file and putting this code in the file organelles.l, allows you to obtain an organelles.l file without the 'Read Only' designation.

c. Once this new L5 code is in the organelles.l file you may either replace the NMOS code with the SCMOS code, or merely comment out the NMOS code. To comment out the NMOS code, simply enclose the NMOS code with a pair of parenthesis containing the word comment inside the left parenthesis, for example:

- (comment
  (NMOS code))

d. Create the cell definition and insert this code before the SCMOS L5 code. The cell definition should be as follows:

- (defun layout-<function>-organelle (drive ratio bit)
  (cond ((= 0 bit)(<name-of-bit0>))
  (t (<name-of-bitn>))))

Replace <function> with either the symbol used for that operator in the input program, or the name of that operator found in the 'function' construct of the library file. For example, the incrementor cell definition used '1+' for the function as shown below:

- defun layout-1+-organelle ....

Replace <name-of-bit0> and <name-of-bitn> with the names used in the defsymbol statement preceding the L5 code for each bit in the organelles.l file. The conditional is required if any bit of the organelle is different than any other bit, and is used to select the correct L5 code for the varying bits.

3. Compile organelles.l creating organelles.o:

a. Before compiling the organelles.l file, move the existing organelles.o file to another file, for example:

- %mv organelles.o organelles.old.o

b. Compile organelles.l:

- %make organelles.o > org.stat &

The & puts the process into the background. A message will be displayed when the job is complete. Check the file org.stat for the status of the compilation once it has completed. Any errors generated during the process will be placed in this file.

4. Make the library changes:

a. The format for the library entry (organelle data structure) is given in Figure D.2 [Ref. 7: p. 25]. The library file entry written for the incrementor organelle is shown in Figure D.3. The three sections of the library entry (See Figure D.2) are the organelle statement (lines 1,2), the gen-form (lines 3-19), and the sim-form (lines 20-24). The organelle statement and the gen-form will require modification; the sim-form should be the same as that used for the NMOS version of the organelle. Again, the NMOS organelle library code may be replaced by the SCMOS code, or commented out.

b. Before making any changes to the library file, copy the existing library file to another file and make a working copy of the library file:

- %mv library library.old
- %cp library.old library.temp

c. Using the vi editor, enter the data for the organelle statement portion of the library entry into a separate file, for example, lib.<filename> The data items are described below:

- <name> - enter the function of the organelle, for example:
  '1+' for the incrementor
- <#control lines> - enter the number of control lines
- <#parameters> - enter the number of inputs to the organelle
- <#testlines> - enter the number of testlines
- result? - enter 'yes' if the organelle provides outputs to the data-path
  - enter 'no' if the organelle provides boolean outputs to the controller

d. After the instantiate command, enter the commands required to place the organelle in the correct position within the data-path. The commands available are listed below:

- first-quadrant - position the origin at the lower left corner
- rotcw - rotate clockwise
- rotccw - rotate counter-clockwise
- mirrorx - fold along the x axis (horizontal axis)
- mirrory - fold along the y axis (vertical axis)

e. Enter the lambda unit values for the connections points. To compute the values of the connection points, make a stipple plot of the organelle. Transfer the <filename>.cif (created in step 1.a) to the CS-VAX via kermit, and execute the following command:

- %Cifplot -s .05 <filename>.cif

When plotting the organelle to obtain the x and y distances for the library entry, use a simple scale value like .05 to make the conversion to lambda units easier.

f. Locate the (0,0) point on the stipple plot. The (0,0) point will be the bottom left hand corner of the organelle, with the inputs entering on the left and the output exiting from the top. Obtain the data for the library entry as described below:

- Measure the y-axis distance (in inches) for the inputs
- Measure the x-axis distance (in inches) for the outputs, carries, test line of most significant bit (MSB), GND and Vdd.
- Measure the length and width (in inches) of the bounding box of the organelle. Note that the total y distance is considered the width and the total x distance is considered the length. Be sure that the length distance specified allows enough clearance of any metal or polysilicon runs or contacts, etc. on the right side of the cell if there are no outputs exiting from this side.

Figure D.4 displays how the measurements were made for the incrementor organelle.

g. Convert these measurements from inches to lambda units using the following formula:

#lambda units =
(distance in inches)/((inches per micron)(microns per lambda))

h. Enter the computed lambda units into the library gen-form, lines 8 through 14 in Figure D.2.

i. The 'test entry, line 15, is used for the location of any test bit sent to the controller from the MSB. In the incrementor, the carry out on the MSB is the test line, and so it has the same x-distance value used for the 'daisy entry of the lower order bits.

j. The 'output-type, line 16, is irrelevant for SCMOS organelles. Ratio is used for NMOS organelles.

k. The 'conductivity value, line 17, is the value used to size Vdd and GND busses for the internal layout [Ref. 7: p. 25].

l. Enter for '#transistors, line 18, the total number of transistors in the organelle.

m. The 'drive item, line 19, is not used for SCMOS, since the organelles have built-in drive.

n. The completed library entry should look similar to the incrementor library entry shown in Figure D.3. Once the library entry for the organelle has been entered into lib.<filename>, write the file and exit the editor.

o. Append this new library entry (lib.<filename>) to the end of the temporary copy and put this into the new library file, using the following command:

- %cat library.temp lib.<filename> > library

Be sure that the file with the new SCMOS entry is called 'library', or MacPitts will not be reading the correct version of the library file when the test circuit is being generated.

5. Using the vi editor, enter the .mac program into your 'macpitts' directory on the ISI workstation. Figure D.5, [Ref. 7: p. 27] shows the format for the .mac file. The items in the input program are described below:

a. For program <name>, line 1, enter the name desired for the completed MacPitts chip layout.

b. The <word-length>, line 1, refers to the number of bits. For example, to test a one-bit layout of the organelle, set <word-length> = 1; to test a multiple-bit layout of the organelle, set <word-length> = n, the number of bits required.

c. Enter the <pin-numbers> for GND, Vdd and the clocks, lines 2 through 6. Be sure to number the GND and Vdd pins as the smallest and largest pin numbers, respectively.

d. Enter the <pin numbers> for the ports, line 7, (input output, i/o and tri-state). The number of pins required for each type of port should be the same as the <word-length> specified in line 1 of the input program.

e. The setq portion of the always construct, line 9, should contain the function the MacPitts chip will perform. For example, the setq portion of the incrementor input program is:

- (setq result (1+ a))

This tells MacPitts to take the input value a, increment it by one and put the result in the output port (result). The MacPitts input programs for the incrementor one-bit and four-bit circuits are shown in Figure D.6.

6. Enter the Macpitts environment and test the connections using the .mac program:

- %Macpitts
- ->(macpitts <.mac filename without extension> hybrid)
  Numerous Heralds and other statements will be displayed as MacPitts runs. After the run is complete, exit the Macpitts environment:
- ->(exit)

The option 'hybrid' must be specified for the MacPitts generation of SCMOS layouts. When generating the NMOS chip layout, the 'nmos' option may be used. However, to be able to plot the completed SCMOS and NMOS chips using the same scale, the NMOS chip should be generated using the 'hybrid' option, so that the chips will be created with the same minimum feature size.

7. To obtain a stipple plot of the MacPitts design, transfer the file (now called filename.cif, where filename here is the same as the .mac filename) to the CS-VAX via kermit. Make sure you have a copy of the file 'patterns' on the CS-VAX in the directory the .cif file is sent to. Execute the following plot command:

- %cifplot -P /path-name to patterns file/patterns -l sn
  -w xmin xmax ymin ymax -b "title of plot" -s .05 filename.cif

The command line options used in this statement [Ref. 10: Cifplot] are described below:

- The -P option allows specification of your own layers and stipple patterns
- The -l sn option (symbolName) eliminates text created by user extension 9
- The -w option allows you to plot a smaller section of the chip. The origin for the window coordinates is the lower left corner of the chip (GND pad at the top). It is a good idea to plot the entire chip (without -w or -s options) the first time, in order to determine the window coordinates.
- The -b "title" prints the title at the top of the plot.

- The -s option specifies the scale of the plot. The number entered specifies the number of inches which represent 1 micron.

The MacPitts generated chip for the incrementor organelle is shown in Figure D.7.

```
(defsymbol incrementorbit0
                nil
                (merge (rect 'CWP 38 85 52 95)
                       (rect 'CWP 38 75 58 85)
                          •
                          •
                          •
                       (mark 'GND! 53 62 'CMS nil)))
(defsymbol incrementorbitn
                nil
                (merge (rect 'CWP 41 82 55 92
                       (rect 'CWP 41 72 61 82)
                          •
                          •
                          •
                       (mark 'cout! 64 90 'CMF nil)))
```

Figure D.1 Sample L5 Code for Incrementor Organelle.

```
(organelle <name> <#control lines> <#parameters>                    1
          <# test lines> result?                                    2
  (lambda(info bit word-length drive ratio)                         3
   (cond                                                            4
    ((eq info 'instantiate)                                         5
     (first-quadrant (mirrorx (mirrory (rotcw                       6
      (layout-<function name>-organelle drive ratio bit))))))       7
    ((eq info 'length) x-distance)                                  8
    ((eq info 'width) y-distance)                                   9
    ((eq info 'inputs) '(y-distance, y-distance, ...)               10
    ((eq info 'output) '(x-distance, x-distance, ...)               11
    ((eq info 'vdd) '(x-distance))                                  12
    ((eq info 'gnd) '(x-distance))                                  13
    ((eq info 'daisy) '(x-distance, x-distance, ...))               14
    ((eq info 'test) '(x-distance))                                 15
    ((eq info 'output-type) '(ratio))                               16
    ((eq info 'conductivity) (quotient # #))                        17
    ((eq info 'number-transistors) '(total # transistors ()))       18
    ((eq info 'drive) '())))                                        19

  (lambda (c a) (list                                               20
   (cond                                                            21
    ((= 0 (*mod (1+ a) (expt 2                                      22
          (get_word_length (env_object env))))) '(1))               23
    (t '(0))) (1+ a))))                                             24
```

Figure D.2 Format of Library File Entry.

```
(organelle 1+ 0 1 1 yes
   (lambda (info bit word-length drive ratio)
    (cond
     ((eq info 'instantiate)
      (first-quadrant (mirrorx (mirrory
       (layout-1+-organelle drive ratio bit)))))
     ((eq info 'width) (cond ((equal 0 bit) 88)
                             (t 85)))
     ((eq info 'length) 44)
     (eq info 'inputs) (cond ((equal 0 bit) '(41))
                             (t '(38))))
     ((eq info 'output) '(26))
     ((eq info 'vdd) '(41))
     ((eq info 'gnd) '(11))
     ((eq info 'daisy) '(4))
     ((eq info 'test) '(4))
     ((eq info 'output-type) '(ratio))
     ((eq info 'conductivity) (quotient 1 0.9574))
     ((eq info 'number-transistors) '(14 ()))
     ((eq info 'drive) '()))))
   (lambda (c a) (list
    (cond
     ((= 0 (*mod (1+ a) (expt 2
        (get_word_length (env-object env))))) '(1))
     (t '(0))) (1+ a))))
```

Figure D.3 Incrementor Library File Entry.

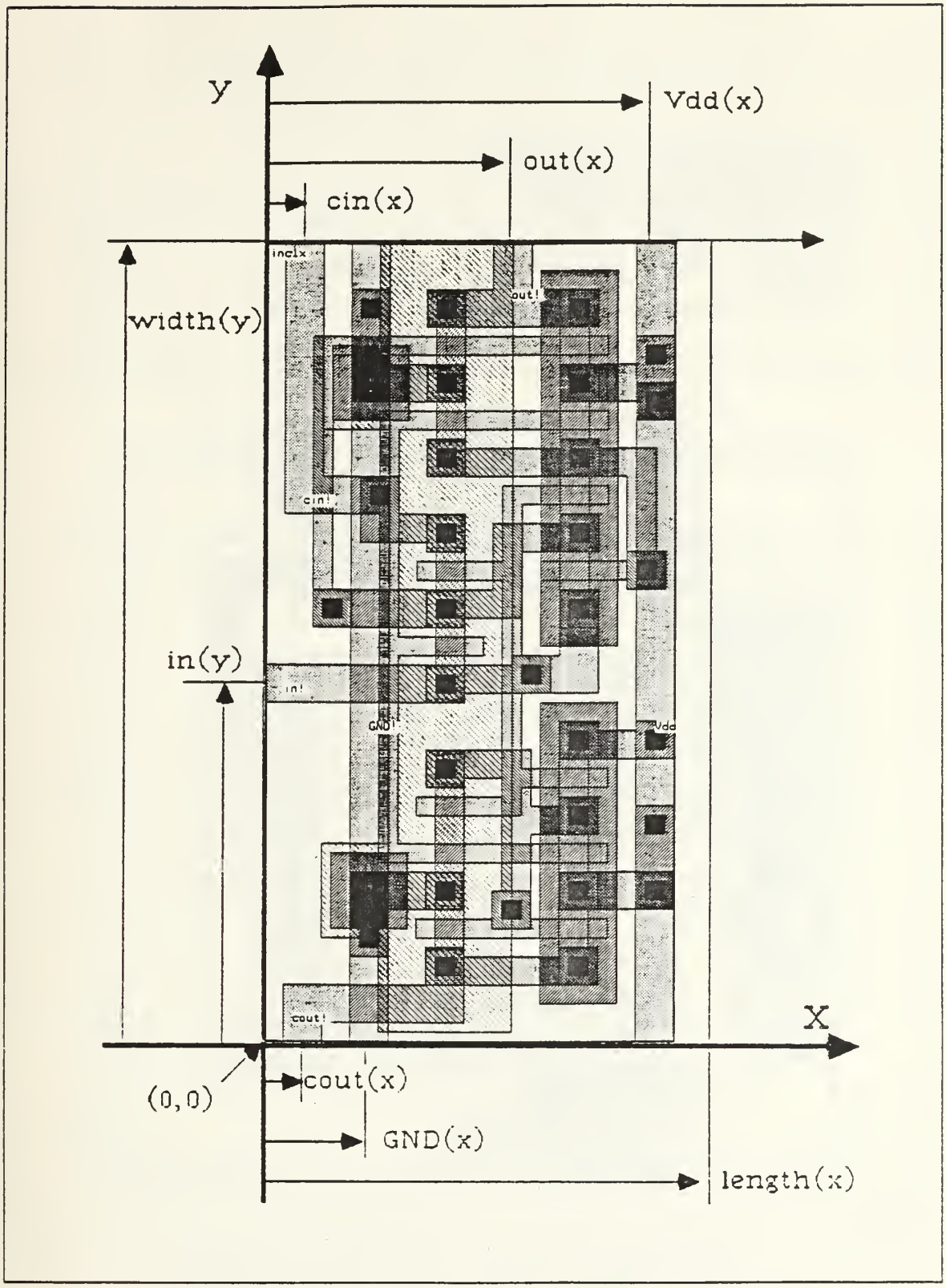

Figure D.4 Connection Point Measurements.

```
(program <name> <word-length>                                    1
  (def <pin-number> ground)                                      2
  (def <pin-number> clock)                                       3
  (def <pin-number> clock)                                       4
  (def <pin-number> clock)                                       5
  (def <pin-number> power                                        6
  (def <port-name> port <type> (<pin-numbers>))                  7
  (always                                                        8
    (setq <port-name> (<function>))))                            9
```

Figure D.5 MacPitts Input Program Format.

One-bit Circuit

```
(program incrementor 1
  (def 1 ground)
  (def a port input (2))
  (def result port output (3))
  (def carry port internal)
  (def 4 phia)
  (def 5 phib)
  (def 6 phic)
  (def 7 power)
  (always
    (setq result (1+ a))))
```

Four-bit Circuit

```
(program incrementor 4
  (def 1 ground)
  (def a port input (2 3 4 5))
  (def result port output (6 7 8 9))
  (def carry port internal)
  (def 10 phia)
  (def 11 phib)
  (def 12 phic)
  (def 13 power)
  (always
    (setq result (1+ a))))
```

Figure D.6 MacPitts Input Programs for the Incrementor.

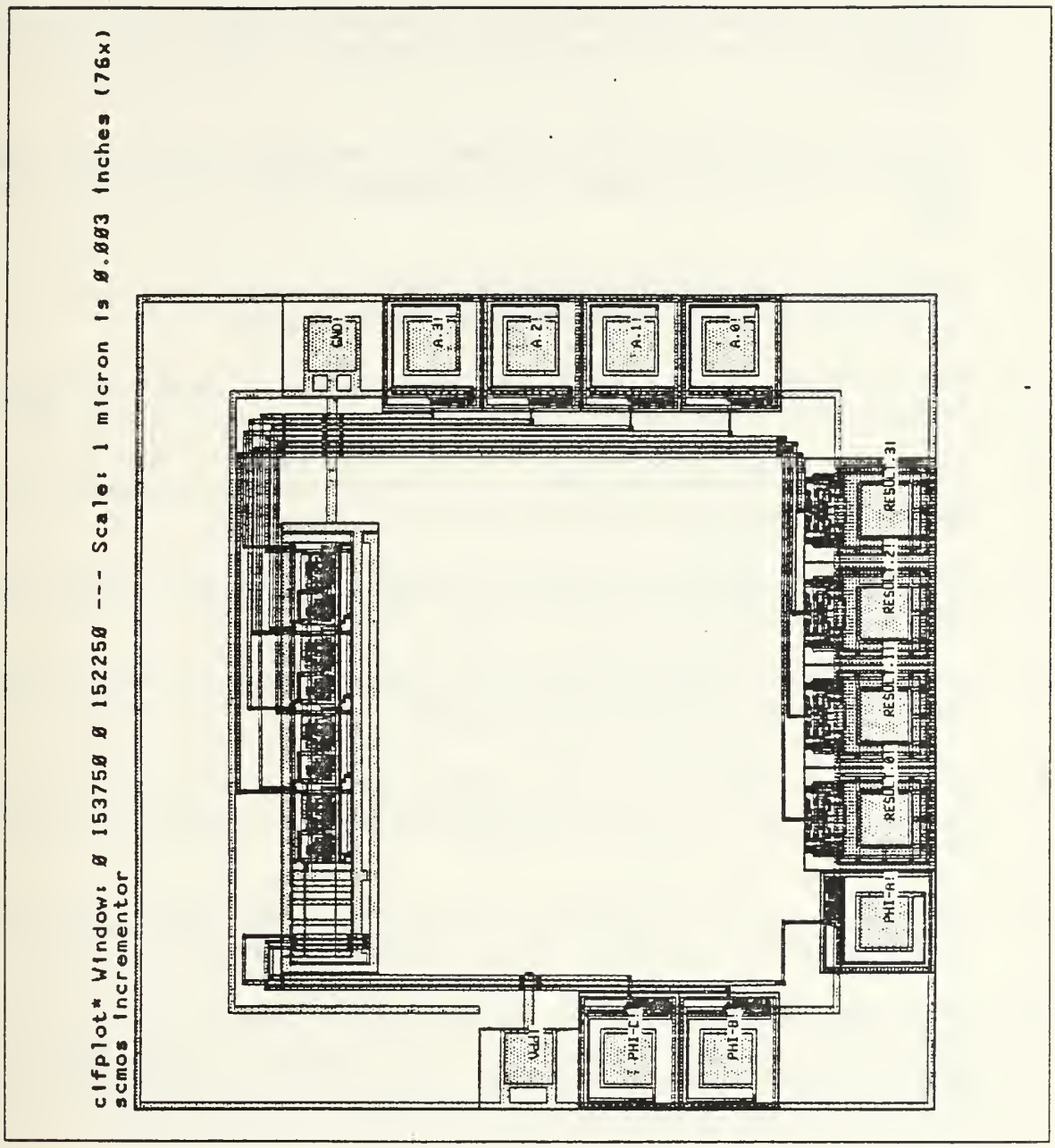

Figure D.7 MacPitts Generated Incrementor Chip.

LIST OF REFERENCES

1. Carlson, D. J., *Application of a Silicon Compiler to VLSI Design of Digital Pipelined Multipliers*, M.S. Thesis, Naval Postgraduate School, Monterey, CA, June 1984.

2. Froede, A. O., *Silicon Compiler Design of Combinational and Pipeline Adder Integrated Circuits*, M.S. Thesis, Naval Postgraduate School, Monterey, CA, June 1985.

3. Larrabee, R. C., *VLSI Design With the MacPitts Silicon Compiler*, M.S. Thesis, Naval Postgraduate School, Monterey, CA, September 1985.

4. Malagon-Fajar, M. A., *Silicon Compilation Using a LISP-Based Layout Language*, M.S. Thesis, Naval Postgraduate School, Monterey, CA, June 1986.

5. Weist, E. L., *A Flowcharting System and Compiler Interface for MacPitts*, M.S. Thesis, Naval Postgraduate School, Monterey, CA, June 1986.

6. Mullarky, A. J., *CMOS Cell Library for a Silicon Compiler*, M.S. Thesis, Naval Postgraduate School, Monterey, CA, March 1987.

7. Malagon, E. G., *Technology Upgrade of a Silicon Compiler*, M.S. Thesis, Naval Postgraduate School, Monterey, CA, June 1987.

8. Harmon, J. E., *Automated Design of a Microprogrammed Controller for a Finite State Machine*, M.S. Thesis, Naval Postgraduate School, Monterey, CA, Work in progress.

9. Southard, J. R., *An Introduction to MacPitts*, Massachusetts Institute of Technology Lincoln Laboratories, Project Report RVLSI-3, 10 February 1983.

10. Scott, Walter S.; Mayo, Robert N.; Hamachi, Gordon; and Ousterhout, John K., *1986 VLSI Tools: Still More Works by the Original Artists*, Report No. UCB/CSD 86/272, Computer Science Division, Electrical Engineering and Computer Sciences, University of California Berkeley, December 1985.

11. Bryant, Randy; Schuster, Mike; and Whiting, Doug, *MOSSIM II : A Switch-Level Simulator for MOS LSI, User's Manual*, 25 January 1983.

12. Vladimirescu, A.; Zhang, Kaihe; Newton, A. R.; Pederson, D. O.; and Sangiovanni-Vincentelli, A., *SPICE User's Guide*, Department of Electrical Engineering and Computer Sciences, University of California Berkeley, October 1983.

13. Weste, Neil H. E., and Eshraghian, Kamran, *Principles of CMOS VLSI Design, A Systems Perspective*, Reading, MA: Addison-Wesley Publishing Co., 1985.

BIBLIOGRAPHY

Ackerman, William B., *How to Design Simulatable CMOS Integrated Circuits*, Department of Electrical Engineering and Computer Science, Massachusetts Institute of Technology, VLSI Memo No. 85-237, March 1985.

Crouch, K. W., *L5 Users's Guide*, Massachusetts Institute of Technology Lincoln Laboratories Project Report RVLSI-5, 7 March 1984.

Gauthier, Richard, *Using the UNIX System*, Reston, VA: Reston Publishing Co. Inc., 1981.

Joy, William, *An Introduction to Display Editing With Vi*, Computer Sciences Division, Department of Electrical and Computer Sciences, University of California Berkeley, 16 September 1980.

Kernighan, Brian W., *A Tutorial Introduction to the UNIX Text Editor*, Bell Laboratories, 21 September 1978.

Shelly, Dale, "Made to Order," *Digital Review*, March 1986.

Wilensky, Robert, *LISPcraft*, New York, NY: W. W. Norton & Co., 1984.

INITIAL DISTRIBUTION LIST

		No. Copies
1.	Defense Technical Information Center Cameron Station Alexandria, VA 22304-6145	2
2.	Library, Code 0142 Naval Postgraduate School Monterey, CA 93943-5002	2
3.	Department Chairman, Code 62 Department of Electrical and Computer Engineering Naval Postgraduate School Monterey, CA 93943-5000	1
4.	Department Chairman, Code 52 Department of Computer Science Naval Postgraduate School Monterey, CA 93943-5000	1
5.	Dr. D. E. Kirk, Code 62KI Department of Electrical and Computer Engineering Naval Postgraduate School Monterey, CA 93943-5000	2
6.	Dr. H. H. Loomis, Jr., Code 62LM Department of Electrical and Computer Engineering Naval Postgraduate School Monterey, CA 93943-5000	2
7.	Dr. C. Yang, Code 62YA Department of Electrical and Computer Engineering Naval Postgraduate School Monterey, CA 93943-5000	2
8.	Dr. M. Zyda, Code 52MZ Department of Computer Science Naval Postgraduate School Monterey, CA 93943-5000	1
9.	Mr. P. Blankenship Massachusetts Institute of Technology Lincoln Laboratory P.O. Box 73 Lexington, MA 02173-0073	1

10. Mr. J. O'Leary — 1
Massachusetts Institute of Technology
Lincoln Laboratory
P.O. Box 73
Lexington, MA 02173-0073

11. Dr. T. Bestul — 1
Naval Research Laboratories
Code 7590
Washington, D.C. 20375

12. Mr. A. DeGroot — 1
Lawrence Livermore National Laboratory
P.O. Box 808
Livermore, CA 94550

13. CAPT E. G. Malagon, USMC — 1
1200 Foursome Lane
Virginia Beach, VA 23462

14. Dr. A. Ross — 1
Naval Research Laboratory, Code 9110
4555 Overlook Ave. SW
Washingotn, D.C. 20375

15. CDR David Southworth — 1
Office of Naval Technology, Code ONT227
800 N. Quincy (BT #1)
Arlington, VA 22217-5000

16. Mr. James Hall — 1
Office of Naval Technology, Code ONT20P4
800 N. Quincy (BT #1)
Arlington, VA 22217-5000

17. LT K. P. Irwin, USN — 1
SMC #2608
Naval Postgraduate School
Monterey, CA 93943

18. LT C. F. Rexach, USN — 1
SMC #2040
Naval Postgraduate School
Monterey, CA 93943

19. CAPT G. R. Steele, USMC — 1
SMC #1888
Naval Postgraduate School
Monterey, CA 93943

20. LT Joan E. Baumstarck, USN — 1
4123 A-1 South 36th Street
Arlington, VA 22206

322 - 898

Thesis
B24293 Baumstarck
c.1 SCMOS silicon compiler organelle design and insertion.

Thesis
B24293 Baumstarck
c.1 SCMOS silicon compiler organelle design and insertion.

AMERICAN BINDERY
TOPEKA KANSAS

thesB24293
SCMOS silicon compiler organelle design
3 2768 000 76740 4
DUDLEY KNOX LIBRARY